Cgins Reference Manual: An Overture Solver for the Incompressible Navier–Stokes Equations on Composite Overlapping Grids,

William D. Henshaw[1]
Centre for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, CA, 94551.

May 23, 2011

LLNL-SM-455871

Abstract:
This is the reference guide for **Cgins**. Cgins is a program that can be used to solve incompressible fluid flow problems in complex geometry in two and three dimensions using composite overlapping grids. It is built upon the **Overture** object-oriented framework. Cgins solves the incompressible Navier-Stokes equations using a pressure-Poisson formulation. Second-order accurate and fourth-order accurate approximations are available. This reference guide describes in some detail the equations being solved, the discrete approximations, time-stepping methods, boundary conditions and convergence results. The reference guide also contains various notes related to different aspects of the equations. The reference guide concludes by providing a collection of interesting computations that have been performed with Cgins.

[1]This work was performed under the auspices of the U.S. Department of Energy (DOE) by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and by DOE contracts from the ASCR Applied Math Program.

Contents

1 Introduction **3**

2 The Equations **3**
2.1 Addition of Temperature and Buoyancy: The Boussinesq Approximation 4
2.2 Axisymmetric Problems 4
2.2.1 The pressure boundary condition for the axisymmetric case 6

3 Discretization **6**

4 Dimensional Units and Non-dimensional Equations **7**

5 Time stepping methods **8**
5.1 Adams Predictor-Corrector 8
5.2 Implicit multistep method with viscous terms implicit 9
5.3 Implicit multistep method with the viscous and non-linear terms implicit 10
5.4 Example linearizations 11
5.5 Variable time stepping 11

6 Divergence Damping **12**
6.1 Artificial Diffusion 12

7 Boundary Conditions **14**

8 Boundary conditions for the fourth-order method **15**

9 Turbulence models **20**
9.1 Wall Shear Stress 20
9.2 Wall Functions 21
9.3 Baldwin-Lomax Zero Equation Model 25
9.4 Spalart-Allmaras turbulence model 25
9.5 $k - \epsilon$ turbulence model 27
9.6 Diffusion Operator 27
9.7 Revised pressure equation 28
9.7.1 Revised pressure boundary condition 29

10 Steady state line solver **31**
10.1 Fourth-order artificial dissipation 32

11 Convergence results **33**

12 Some sample simulations **36**
12.1 Falling cylinders in an incompressible flow 36
12.2 Flow past two cylinders 37
12.3 Pitching and plunging airofil 38
12.4 Fluttering plate 39
12.5 Simulation of flow past a blood clot filter 40
12.6 Incompressible flow past a truck 41
12.7 Incompressible flow past a city scape 42

1 Introduction

This is the reference guide for **Cgins**. Cgins is a program that can be used to solve incompressible fluid flow problems in complex geometry in two and three dimensions using composite overlapping grids. It is built upon the **Overture** object-oriented framework [1],[4],[2]. Cgins solves the incompressible Navier-Stokes equations using a pressure-Poisson formulation. Second-order accurate and fourth-order accurate approximations are available. Cgins can be used to

- solve problems in two and three dimensional complex domains,
- solve problems on moving grids (specified motion and rigid body motion),
- solve temperature dependent flows with buoyancy using the Boussinesq approximation,
- solve axisymmetric flows.

This reference guide describes in some detail the equations being solved, the discrete approximations, time-stepping methods, boundary conditions and convergence results. The reference guide also contains various notes related to different aspects of the equations. The reference guide concludes by providing a collection of interesting computations that have been performed with Cgins.
Other documents of interest that are available through the **Overture** home page are

- The Cgins User Guide [9] for getting started with Cgins and for descriptions of the various run-time options and parameters.
- The overlapping grid generator, `Ogen`, [6]. Use this program to make grids for cgins.
- Mapping class documentation : `mapping.tex`, [5]. Many of the mappings that are used to create an overlapping grid are documented here.
- Interactive plotting : `PlotStuff.tex`, [8].
- `Oges` overlapping grid equation solver, used by cgins to solve implicit time stepping equations and the Poisson equation for the pressure, [7].

2 The Equations

The incompressible Navier-Stokes equations are

$$\mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \nu\Delta\mathbf{u}, \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0. \tag{2}$$

We solve the incompressible Navier-Stokes equations written in the form (pressure-poisson system)

$$\left.\begin{aligned} \mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p - \nu\Delta\mathbf{u} - \mathbf{f} &= 0 \\ \Delta p - (\nabla u\cdot\mathbf{u}_x + \nabla v\cdot\mathbf{u}_y + \nabla w\cdot\mathbf{u}_z) - C_d(\nu)\nabla\cdot\mathbf{u} - \nabla\cdot\mathbf{f} &= 0 \end{aligned}\right\} \quad \mathbf{x}\in\Omega \tag{3}$$

$$\left.\begin{aligned} B(\mathbf{u},p) &= 0 \\ \nabla\cdot\mathbf{u} &= 0 \end{aligned}\right\} \quad \mathbf{x}\in\partial\Omega$$

$$\mathbf{u}(\mathbf{x},0) = \mathbf{u}_0(\mathbf{x}) \qquad \text{at } t=0$$

There are n_d boundary conditions, $B(\mathbf{u},p)=0$, where n_d is the number of space dimensions. On a no-slip wall, for example, $\mathbf{u}=0$. In addition, a boundary condition is required for the pressure. The boundary condition $\nabla\cdot\mathbf{u}=0$ is added. With this extra boundary condition it follows that the above problem is equivalent to the formulation with the Poisson equation for the pressure replaced by $\nabla\cdot\mathbf{u}=0$ everywhere. The term $C_d(\nu)\nabla\cdot\mathbf{u}$ appearing in the equation for the pressure is used to damp the divergence [10]. For further details see also [3]

2.1 Addition of Temperature and Buoyancy: The Boussinesq Approximation

The effects of temperature and buoyancy can be modeled with the Boussinesq approximation given by

$$\begin{aligned}
\mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p - \nu\Delta\mathbf{u} + \alpha\mathbf{g}T - \mathbf{f} &= 0,\\
\Delta p - \mathcal{J}(\nabla\mathbf{u}) - \alpha(\mathbf{g}\cdot\nabla)T - \nabla\cdot\mathbf{f} &= 0,\\
T_t + (\mathbf{u}\cdot\nabla)T - k\Delta T - f_T &= 0,\\
\mathcal{J}(\nabla\mathbf{u}) \equiv (\nabla u\cdot\mathbf{u}_x + \nabla v\cdot\mathbf{u}_y + \nabla w\cdot\mathbf{u}_z)&\ .
\end{aligned}$$

Here T represents a temperature perturbation, $\mathbf{g}$ is the acceleration due to gravity, α is the coefficient of thermal expansivity and k is a coefficient of thermal conductivity, $k = \lambda/(\rho C_p)$.

The Rayleigh number and Prandtl number are defined as

$$Ra = \alpha g \Delta T d^3/(\nu\kappa), \qquad Pr = \nu/\kappa$$

where ΔT is the representative temperature difference and d is a length scale.

2.2 Axisymmetric Problems

Here I describe the equations that are solved for axisymmetric problems.

Let the cylindrical coordinates be (x, r, θ) where x is the axial variable, r the radial variable and θ is the azimuthal angle about the axis $r = 0$. Let the velocity be $\mathbf{u} = U\hat{\mathbf{x}} + V\hat{\mathbf{r}} + W\hat{\boldsymbol{\theta}}$ where (U, V, W) are the components of axial, radial and azimuthal velocities and $(\hat{\mathbf{x}}, \hat{\mathbf{r}}, \hat{\boldsymbol{\theta}})$ are the three unit vectors in the coordinate directions.

In cylindrical coordinates we have the general relations

$$\begin{aligned}
\mathbf{F} &= \hat{\mathbf{x}}F^x + \hat{\mathbf{r}}F^r + \hat{\boldsymbol{\theta}}F^\theta\\
\nabla V &= \hat{\mathbf{x}}V_x + \hat{\mathbf{r}}V_r + \frac{\hat{\boldsymbol{\theta}}}{r}V_\theta\\
\mathbf{n}\cdot\nabla\mathbf{F} &= \hat{\mathbf{x}}(\mathbf{n}\cdot\nabla F^x) + \hat{\mathbf{r}}(\mathbf{n}\cdot\nabla F^r - \frac{n^\theta F^\theta}{r}) + \hat{\boldsymbol{\theta}}(\mathbf{n}\cdot\nabla F^\theta + \frac{n^\theta F^r}{r})\\
\nabla\cdot\mathbf{F} &= F^x_x + \frac{1}{r}(rF^r)_r + \frac{1}{r}F^\theta_\theta\\
\Delta V &= V_{xx} + \frac{1}{r}(rV_r)_r + \frac{1}{r^2}V_{\theta\theta}\\
\Delta\mathbf{F} &= \hat{\mathbf{x}}(\Delta F^x) + \hat{\mathbf{r}}(\Delta F^r - \frac{F^r}{r^2} - \frac{2}{r^2}F^\theta_\theta) + \hat{\boldsymbol{\theta}}(\Delta F^\theta + \frac{2}{r^2}F^r_\theta - \frac{F^\theta}{r^2})\\
\nabla\times\mathbf{F} &= \hat{\mathbf{x}}(\frac{1}{r}(rF^\theta)_r - \frac{1}{r}F^r_\theta) + \hat{\mathbf{r}}(\frac{1}{r}F^x_\theta - F^\theta_x) + \hat{\boldsymbol{\theta}}(F^r_x - F^x_r)
\end{aligned}$$

The incompressible Navier-Stokes equations in cylindrical coordinates are (see Batchelor)

$$\begin{aligned}
U_t + UU_x + VU_r + \frac{W}{r}U_\theta + p_x &= \nu(U_{xx} + \frac{1}{r}(rU_r)_r + \frac{1}{r^2}U_{\theta\theta})\\
V_t + UV_x + VV_r + \frac{W}{r}V_\theta - \frac{W^2}{r} + p_r &= \nu(V_{xx} + \frac{1}{r}(rV_r)_r + \frac{1}{r^2}V_{\theta\theta} - \frac{V}{r^2} - \frac{2}{r^2}W_\theta)\\
W_t + UW_x + VW_r + \frac{W}{r}W_\theta + \frac{VW}{r} + \frac{1}{r}p_\theta &= \nu(W_{xx} + \frac{1}{r}(rW_r)_r + \frac{1}{r^2}W_{\theta\theta} - \frac{W}{r^2} + \frac{2}{r^2}V_\theta)\\
U_x + \frac{1}{r}(rV)_r + \frac{1}{r}W_{\theta\theta} &= 0
\end{aligned}$$

For axisymmetric problems with no swirl, $W = 0$ and all derivatives with respect to θ are zero,

$$U_t + UU_x + VU_r + p_x = \nu(U_{xx} + \frac{1}{r}(rU_r)_r \tag{4}$$

$$V_t + UV_x + VV_r + p_r = \nu(V_{xx} + \frac{1}{r}(rV_r)_r - \frac{V}{r^2}) \tag{5}$$

$$U_x + \frac{1}{r}(rV)_r = 0 \tag{6}$$

The divergence of the advection terms is

$$\begin{aligned}\nabla \cdot (UU_x + VU_r, UV_x + VV_r) &= (UU_x + VU_r)_x + \frac{1}{r}[r(UV_x + VV_r)]_r \\ &= U_x^2 + 2V_xU_r + V_r^2 + U\{(U_x)_x + \frac{1}{r}[r(V_x)_r]\} + V\{(U_r)_x + \frac{1}{r}[r(V_r)]_r\} \\ &= U_x^2 + 2V_xU_r + V_r^2 + U(\nabla \cdot \mathbf{U})_x + V(\nabla \cdot \mathbf{U})_r\end{aligned}$$

and thus pressure equation becomes

$$p_{xx} + \frac{1}{r}(rp_r)_r = U_x^2 + 2V_xU_r + V_r^2$$

The boundary conditions on the axis of symmetry are (Note that although the normal component of the velocity usually has even symmetry, $V = \mathbf{U} \cdot \hat{r}$ will have even symmetry at $r = 0$ since $\hat{r}$ flips sign as we cross the axis)

$$\begin{aligned} U_r(x,0) &= 0 \\ V(x,0) &= 0 \quad , \quad V_r(x,0) = 0 \\ p_r(x,0) &= 0 \end{aligned}$$

In general all odd derivatives of $\mathbf{U}$ and p with respect to r will be zero at $r = 0$.

Note that for r small,

$$\begin{aligned}\frac{1}{r}(rV_r)_r - \frac{V}{r^2} &= V_{rr} + \frac{1}{r}V_r - \frac{V}{r^2} \\ &= V_{rr} + \frac{1}{r}(V_r(x,0) + rV_{rr}(x,0) + O(r^2)) - \frac{1}{r^2}(V(x,0) + rV_r(x,0) + \frac{r^2}{2}V_{rr}(x,0) + O(r^3) \\ &= \frac{3}{2}V_{rr}(x,0) + O(r) \\ &= O(r)\end{aligned}$$

and

$$\frac{1}{r}(rU_r)_r = U_{rr} + \frac{1}{r}U_r = 2U_{rr} + O(r)$$

We can use these last two results to evaluate the viscous terms on the boundary $r = 0$, (eliminating the removable singularity) although the dirichlet condition $V(x,0) = 0$ obviates the need for the former equation on the boundary.

Note that in inviscid flow the only difference between the axi-symmetric equations and the 2D equations is a change to the pressure equation (or to the incompressibility equation) with the addition of the $\frac{1}{r}p_r$ term. With viscosity there are also differences in the viscous terms.

2.2.1 The pressure boundary condition for the axisymmetric case

The pressure boundary condition is formed from the normal component of the momentum equations. For a no-slip wall this becomes

$$p_n = \nu n_x\Big(U_{xx} + \frac{1}{r}(rU_r)_r\Big) + \nu n_r\Big(V_{xx} + V_{rr} + \frac{1}{r}V_r - \frac{V}{r^2}\Big)$$

We can eliminate some of the normal derivatives in this last expression (forming the equivalent of the *curl-curl* boundary conditions described in []) by taking derivative of the expression (6) for the divergence,

$$U_x + V_r + \frac{1}{r}V = 0$$

giving

$$U_{xx} = -V_{xr} - \frac{1}{r}V_x = 0$$
$$V_{rr} + \frac{1}{r}V_r - \frac{1}{r^2}V = -U_{xr}$$

This leads to the pressure boundary condition

$$p_n = \nu n_x\Big(- V_{xr} + U_{rr} - \frac{1}{r}V_x + \frac{1}{r}U_r\Big) + \nu n_r\Big(V_{xx} - U_{xr}\Big) \tag{7}$$

On the axis, $r = 0$, this becomes

$$p_n = 2\nu n_x\Big(- V_{xr} + U_{rr}\Big) + \nu n_r\Big(V_{xx} - U_{xr}\Big), \qquad (\text{for } r = 0).$$

3 Discretization

Let $\mathbf{V}_i$ and P_i denote the discrete approximations to $\mathbf{u}$ and p so that

$$\mathbf{V}_i \approx \mathbf{u}(\mathbf{x}_i) \quad , \quad P_i \approx p(\mathbf{x}_i) \ .$$

Here $\mathbf{V}_i = (V_{1i}, V_{2i}, V_{3i})$ and $i = (i_1, i_2, i_3)$ is a multi-index. After discretizing in space the equations we solve are of the form

$$\left.\begin{array}{rcl} \frac{d}{dt}\mathbf{V}_i + (\mathbf{V}_i\cdot\nabla_h)\mathbf{V}_i + \nabla_h P_i - \nu\Delta_h\mathbf{V}_i - \mathbf{f}(\mathbf{x}_i,t) &=& 0 \\ \Delta_h P_i - \sum_m \nabla_h V_{m,i}\cdot D_{m,h}\mathbf{V}_i - C_{d,i}\nabla_h\cdot\mathbf{V}_i - \nabla_h\cdot\mathbf{f}(\mathbf{x}_i,t) &=& 0 \end{array}\right\} \quad \mathbf{x}\in\Omega$$

$$\left.\begin{array}{rcl} B(\mathbf{V}_i, P_i) &=& 0 \\ \nabla_h\cdot\mathbf{V}_i &=& 0 \end{array}\right\} \quad \mathbf{x}_i \in \partial\Omega_h$$

$$\mathbf{V}(\mathbf{x}_i, 0) = \mathbf{U}_0(\mathbf{x}_i) \qquad \text{at } t = 0$$

where the divergence damping coefficient, $C_{d,i}$ is defined below. The subscript "h" denotes a second or fourth-order centred difference approximation,

$$D_{m,h} \approx \frac{\partial}{\partial x_m} \quad , \quad \nabla_h = (D_{1,h}, D_{2,h}, D_{3,h}) \quad , \quad \Delta_h \approx \sum_m \frac{\partial^2}{\partial x_m^2}$$

Extra numerical boundary conditions are also added, see [3] [13] for further details. An artificial diffusion term can be added to the momentum equations. This is described in section (6.1).

When discretized in space on an overlapping grid this system of PDEs can be thought of as a large system of ODEs of the form

$$\frac{d\mathbf{U}}{dt} = \mathcal{F}(t, \mathbf{U}, \mathbf{P})$$

where $\mathbf{U}$ is a vector of all solution values at all grid points. For the purpose of discussing time-stepping methods it is often convenient to think of the pressure as simply a function of $\mathbf{U}$, $\mathbf{P} = \mathcal{P}(\mathbf{U})$ There are also interpolation equations that need to be satisfied but this causes no difficulties.

4 Dimensional Units and Non-dimensional Equations

When you solve the incompressible Navier-Stokes equations with Cgins it solves the following equations

$$\mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \nu\Delta\mathbf{u}, \tag{8}$$

$$\nabla\cdot\mathbf{u} = 0. \tag{9}$$

There is only one parameter that you specify and that is ν, the kinematic viscosity. Cgins does NOT scale the input values that you specify for the initial conditions and boundary conditions. Cgins does not scale the dimensions of the grid. Also note that the pressure p is only determined up to a constant.

Let's relate these equations to the equations in dimensional form,

$$\mathbf{u}^d_{t^d} + (\mathbf{u}^d\cdot\nabla^d)\mathbf{u}^d + \frac{1}{\rho^d}\nabla^d p^d = \frac{\mu}{\rho^d}\Delta^d\mathbf{u}^d, \tag{10}$$

$$\nabla^d\cdot\mathbf{u}^d = 0. \tag{11}$$

Here the density ρ^d is a constant and μ is the constant viscosity. The superscript d denotes dimensional units. For example, the components of $\mathbf{u}^d$ might be measured in m/s, lengths in m, the pressure p^d in N/m^2 and the viscosity coefficient μ in $kg/(m\ s)$.

Suppose that you want to use dimensional units, for example MKS units, to specify a problem for Cgins. The grid should then be built with dimensions in m. The inflow values for the velocity should be given in m/s. The kinematic viscosity should be set equal to $\nu = \mu/\rho^d$ which has units m^2/s. Comparing equation (10) to equation (8) we see that the pressure computed by Cgins will be $p = p^d/\rho^d$. Thus you should specify boundary conditions and initial conditions for p from $p = p^d/\rho^d$ (which has units of m^2/s^2).

In general one may want to non-dimensionalize the equations before solving them with Cgins. Let us choose appropriate values, U, L, R, to scale the length, velocity and density, and define non-dimensional variables

$$\mathbf{x}^n = \mathbf{x}^d/L\ , \qquad \rho^n = \rho^d/R\ , \tag{12}$$

$$t^n = t^d/(L/U)\ , \qquad \mathbf{u}^n = \mathbf{u}^d/U\ , \tag{13}$$

$$p^n = p^d/(RU^2)\ . \tag{14}$$

The Navier-Stokes equations then become

$$\mathbf{u}^n_{t^n} + (\mathbf{u}^n\cdot\nabla^n)\mathbf{u}^n + \frac{1}{\rho^n}\nabla^n p^n = \frac{\mu}{\rho^n RUL}\Delta^n\mathbf{u}^n, \tag{15}$$

$$\nabla^n\cdot\mathbf{u}^n = 0. \tag{16}$$

In this case we will generate the grid with lengths that have been scaled by L. We will specify inflow velocities with values scaled by U, the pressure will be scaled by RU^2, the kinematic viscosity will be defined by $\nu = \mu/(\rho^n RUL)$ and the non-dimensional pressure p^n will be related to the computed value p by $p = p^n/\rho^n$.

5 Time stepping methods

5.1 Adams Predictor-Corrector

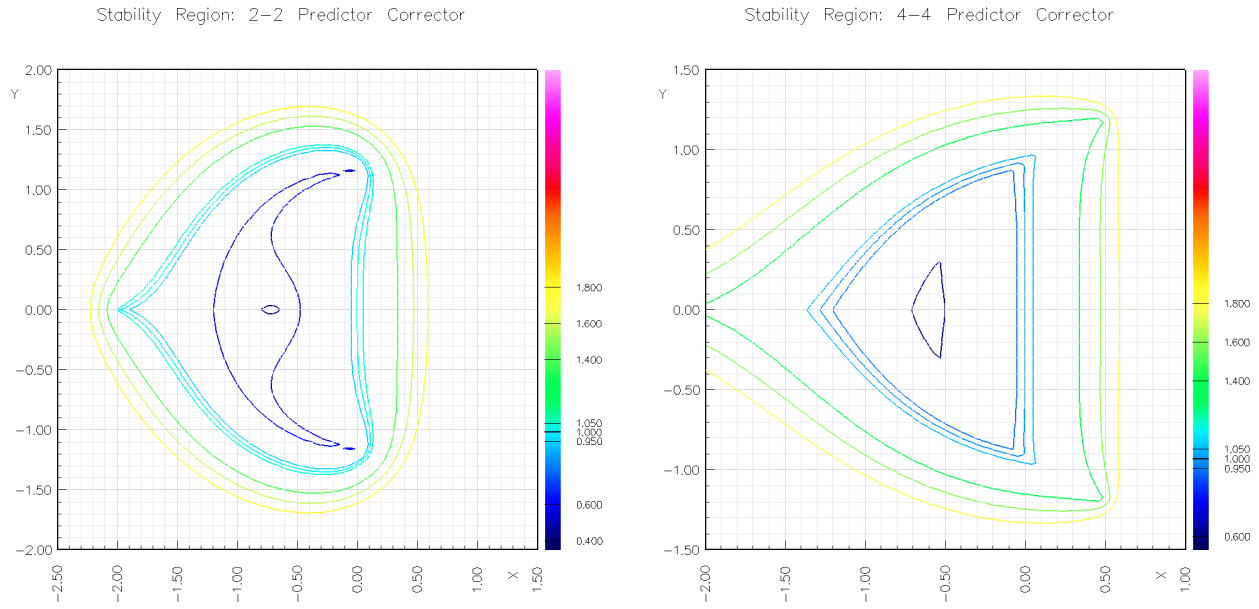


Figure 1: Stability regions for the predictor corrector methods. Left: second-order method, PECE mode. Right: fourth-order method, PECE mode.

If we write the INS equations as

$$\mathbf{u}_t = \mathbf{f}(\mathbf{u}, p)$$

where we may think of the pressure p as simply a function of $\mathbf{u}$ and we may use any time integrator in a method-of-lines fashion.

The **second-order accurate Adams predictor-corrector** time stepping method for the INS equations can be chosen with the "`adams PC`" option. It is defined by

$$\begin{aligned}\frac{\mathbf{u}^p - \mathbf{u}^n}{\Delta t} &= \frac{3}{2}\mathbf{f}^n - \frac{1}{2}\mathbf{f}^{n-1} \\ \frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} &= \frac{1}{2}\mathbf{f}^p + \frac{1}{2}\mathbf{f}^n\end{aligned}$$

where we have shown one correction step (one may optionally correct more than one time). The stability region for this method is shown in figure (1).

To allow for a time-step that may change we actually use

$$\begin{aligned}\frac{\mathbf{u}^p - \mathbf{u}^n}{\Delta t} &= p_0\mathbf{f}^n + p_1\mathbf{f}^{n-1} \\ \frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} &= \frac{1}{2}\mathbf{f}^p + \frac{1}{2}\mathbf{f}^n \\ p_0 &= 1 + \Delta t/(2\Delta t_1) \\ p_1 &= -\Delta t/(2\Delta t_1)\end{aligned}$$

where $\Delta t_1 = t_n - t_n - 1$.

The **fourth-order accurate Adams predictor-corrector** time stepping method for the INS equations can be chosen with the "`adams PC order 4`" option. It is defined by

$$\frac{\mathbf{u}^p - \mathbf{u}^n}{\Delta t} = \frac{1}{24}\Big[55\mathbf{f}^n - 59\mathbf{f}^{n-1} + 37\mathbf{f}_{n-2} - 9\mathbf{f}_{n-3}\Big]$$
$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = \frac{1}{24}\Big[9\mathbf{f}^p + 19\mathbf{f}^n - 5\mathbf{f}^{n-1} + \mathbf{f}_{n-2}\Big]$$

(see for example Lambert[**?**]). The stability region for this method is shown in figure (1).

To allow for a time-step that may change we actually use

$$\frac{\mathbf{u}^p - \mathbf{u}^n}{\Delta t} = p_0\mathbf{f}^n + p_1\mathbf{f}^{n-1} + p_2\mathbf{f}_{n-2} + p_3\mathbf{f}_{n-3}$$
$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = c_0\mathbf{f}^p + c_1\mathbf{f}^n + c_2\mathbf{f}^{n-1} + c_3\mathbf{f}_{n-2}$$
$$\begin{aligned}
p_0 =&(6\Delta t_0\Delta t_2\Delta t_2 + 12\Delta t_2\Delta t_2\Delta t_1 + 8\Delta t_0\Delta t_0\Delta t_2 + 24\Delta t_2\Delta t_0\Delta t_1 + \\
&12\Delta t_2\Delta t_1\Delta t_3 + 6\Delta t_3\Delta t_2\Delta t_0 + 24\Delta t_1\Delta t_1\Delta t_2 + 12\Delta t_0\Delta t_3\Delta t_1 + 18\Delta t_0\Delta t_1 \\
&\Delta t_1 + 4\Delta t_0\Delta t_0\Delta t_3 + 12\Delta t_1\Delta t_1\Delta t_3 + 3\Delta t_0\Delta t_0\Delta t_0 + 12\Delta t_0\Delta t_0\Delta t_1 + 12\Delta t_1 \\
&\Delta t_1\Delta t_1)\Delta t_0/(\Delta t_1 + \Delta t_2 + \Delta t_3)/\Delta t_1/(\Delta t_1 + \Delta t_2)/12 \\
p_1 =&-\Delta t_0\Delta t_0(6\Delta t_1\Delta t_1 + 6\Delta t_3\Delta t_1 + 12\Delta t_2\Delta t_1 + 8\Delta t_0\Delta t_1 + 3\Delta t_0 \\
&\Delta t_0 + 6\Delta t_2\Delta t_3 + 4\Delta t_0\Delta t_3 + 8\Delta t_2\Delta t_0 + 6\Delta t_2\Delta t_2)/\Delta t_1/(\Delta t_2 + \Delta t_3)/\Delta t_2/12 \\
p_2 =&\Delta t_0\Delta t_0(6\Delta t_1\Delta t_1 + 6\Delta t_2\Delta t_1 + 6\Delta t_3\Delta t_1 + 8\Delta t_0\Delta t_1 + 3\Delta t_0\Delta t_0 \\
&+ 4\Delta t_2\Delta t_0 + 4\Delta t_0\Delta t_3)/\Delta t_3/\Delta t_2/(\Delta t_1 + \Delta t_2)/12 \\
p_3 =&-(6\Delta t_1\Delta t_1 + 6\Delta t_2\Delta t_1 + 8\Delta t_0\Delta t_1 + 4\Delta t_2\Delta t_0 + 3\Delta t_0\Delta t_0)\Delta t_0 \\
&\Delta t_0/(\Delta t_1 + \Delta t_2 + \Delta t_3)/(\Delta t_2 + \Delta t_3)/\Delta t_3/12 \\
c_0 =&(6\Delta t_1\Delta t_1 + 6\Delta t_2\Delta t_1 + 8\Delta t_0\Delta t_1 + 4\Delta t_2\Delta t_0 + 3\Delta t_0\Delta t_0) \\
&\Delta t_0/(\Delta t_0 + \Delta t_1 + \Delta t_2)/(\Delta t_0 + \Delta t_1)/12 \\
c_1 =&\Delta t_0(\Delta t_0\Delta t_0 + 4\Delta t_0\Delta t_1 + 2\Delta t_2\Delta t_0 + 6\Delta t_1\Delta t_1 + 6\Delta t_2\Delta t_1)/(\Delta t_1 + \Delta t_2)/\Delta t_1/12 \\
c_2 =&-\Delta t_0\Delta t_0\Delta t_0(\Delta t_0 + 2\Delta t_1 + 2\Delta t_2)/(\Delta t_0 + \Delta t_1)/\Delta t_2/\Delta t_1/12 \\
c_3 =&(\Delta t_0 + 2\Delta t_1)\Delta t_0\Delta t_0\Delta t_0/(\Delta t_0 + \Delta t_1 + \Delta t_2)/(\Delta t_1 + \Delta t_2)/\Delta t_2/12
\end{aligned}$$

where $\Delta t_m = t_{n+1-m} - t_{n-m}$, $m = 0, 1, 2, 3$.

5.2 Implicit multistep method with viscous terms implicit

With the `implicit` time stepping method the INS equations are integrated with the viscous terms treated implicitly and the other terms treated with a 2nd-order Adams predictor corrector. If we split the equations into an explicit and implicit part,

$$\begin{aligned}
\mathbf{u}_t &= [-(\mathbf{u}\cdot\nabla)\mathbf{u} - \nabla p] + \nu\Delta\mathbf{u} \\
\mathbf{u}_t &= \mathbf{f}_E + A\mathbf{u} \\
\mathbf{f}_E &= -(\mathbf{u}\cdot\nabla)\mathbf{u} - \nabla p \\
A\mathbf{u} &= \nu\Delta\mathbf{u}
\end{aligned}$$

then the time step consists of a predictor,

$$\frac{\mathbf{u}^p - \mathbf{u}^n}{\Delta t} = \frac{3}{2}\mathbf{f}_E^n - \frac{1}{2}\mathbf{f}_E^{n-1} + \alpha A\mathbf{u}^p + (1-\alpha)A\mathbf{u}^n$$

and a corrector

$$\frac{\mathbf{u}^c - \mathbf{u}^n}{\Delta t} = \frac{1}{2}\mathbf{f}_E^p + \frac{1}{2}\mathbf{f}_E^n + \alpha A\mathbf{u}^c + (1-\alpha)A\mathbf{u}^n$$

The `implicit factor` α can be set as a parameter. A value of $\alpha = \frac{1}{2}$ will give a second-order Crank-Nicolson method. A value of $\alpha = 1$ will give a first-order backward-Euler method.

5.3 Implicit multistep method with the viscous and non-linear terms implicit

Consider now the case of treating the viscous and non-linear terms in an implicit manner.

In the general case we are solving a nonlinear equation of the form

$$\mathbf{u}_t = \mathbf{f}(\mathbf{u}).$$

We linearize a part of the operator $\mathbf{f}(\mathbf{u})$ around the state $\mathbf{u}^*(\mathbf{x}, t)$ and write the equation as

$$\begin{aligned} \mathbf{u}_t &= L(\mathbf{u};\mathbf{u}^*) + \big(\mathbf{f}(\mathbf{u}) - L(\mathbf{u};\mathbf{u}^*)\big) && (17)\\ &= L(\mathbf{u};\mathbf{u}^*) + \mathbf{f}_E(\mathbf{u};\mathbf{u}^*) && (18)\end{aligned}$$

where

$$\mathbf{f}_E(\mathbf{u};\mathbf{u}^*) \equiv \mathbf{f}(\mathbf{u}) - L(\mathbf{u};\mathbf{u}^*) \tag{19}$$

Here $L(\mathbf{u};\mathbf{u}^*)$ is a linear operator of $\mathbf{u}$ that depends upon $\mathbf{u}^*$. For example if $\mathbf{f}(\mathbf{u}) = (\mathbf{u}\cdot\nabla)\mathbf{u}$ then one choice could be $L(\mathbf{u};\mathbf{u}^*) = (\mathbf{u}^*\cdot\nabla)\mathbf{u}$. Another choice could be $L(\mathbf{u};\mathbf{u}^*) = (\mathbf{u}^*\cdot\nabla)\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u}^*$. We can now define a implicit time-stepping method that uses the form (18). A backward-Euler approximation to the full equations is

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t} = L(\mathbf{u}^{n+1};\mathbf{u}^*) + \mathbf{f}_E(\mathbf{u}^{n+1};\mathbf{u}^*)\ .$$

This equation can be solved by *implicit* iteration (treating the L term implicitly),

$$\frac{\mathbf{v}^{k}-\mathbf{u}^n}{\Delta t} = L(\mathbf{v}^{k};\mathbf{u}^*) + \mathbf{f}_E(\mathbf{v}^{k-1};\mathbf{u}^*)$$

where we take $\mathbf{v}^0 = \mathbf{u}^n$ and iterate on $k = 0, 1, 2, ...$ for some number of steps. If we iterate until convergence then we will have solved the original equations with backward-Euler, independent of the choice L and $\mathbf{u}^*$,

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t} = \mathbf{f}(\mathbf{u}^{n+1})\ .$$

Of course for best convergence for a large value of Δt, we want make good choices for L and $\mathbf{u}^*$.

We could also solve the equations with the trapezodial like rule

$$\begin{aligned}\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t} &= \alpha\mathbf{f}(\mathbf{u}^{n+1}) + (1-\alpha)\mathbf{f}(\mathbf{u}^n)\\ &= \alpha L(\mathbf{u}^{n+1};\mathbf{u}^*) + (1-\alpha)L(\mathbf{u}^n;\mathbf{u}^*) + \alpha\mathbf{f}_E(\mathbf{u}^{n+1};\mathbf{u}^*) + (1-\alpha)\mathbf{f}_E(\mathbf{u}^n;\mathbf{u}^*),\\ &= \alpha L(\mathbf{u}^{n+1};\mathbf{u}^*) + \alpha\mathbf{f}_E(\mathbf{u}^{n+1};\mathbf{u}^*) + (1-\alpha)\mathbf{f}(\mathbf{u}^n)\\ &= \alpha L(\mathbf{u}^{n+1};\mathbf{u}^*) + \alpha\big(\mathbf{f}(\mathbf{u}^{n+1}) - L(\mathbf{u}^{n+1};\mathbf{u}^*)\big) + (1-\alpha)\mathbf{f}(\mathbf{u}^n)\end{aligned}$$

where $\alpha = \frac{1}{2}$ is the trapezodial rule and $\alpha = 1$ is backward-Euler. This equation can be solved by the implicit iteration

$$\frac{\mathbf{v}^{k}-\mathbf{u}^n}{\Delta t} = \alpha L(\mathbf{v}^{k};\mathbf{u}^*) + \alpha\big(\mathbf{f}(\mathbf{v}^{k-1}) - L(\mathbf{v}^{k-1};\mathbf{u}^*)\big) + (1-\alpha)\mathbf{f}(\mathbf{u}^n)$$

For $\alpha = 1/2$, the first iterate $\mathbf{v}^1$ is second-order accurate in the implicit part L and first order accurate in the explicit part $\mathbf{f}_E$. If $\mathbf{f}_E(\mathbf{u}^n;\mathbf{u}^*) = \mathcal{O}(\Delta t)$ then the $\mathbf{v}^1$ is second-order accurate. For $\alpha = 1/2$ the second and later iterates, $\mathbf{v}^k$, $k \geq 2$, should be second-order accurate.

To implement this scheme we need to (1) build the matrix M for the implicit operator,

$$M\mathbf{v}^k = (I - \alpha\Delta t A(\mathbf{u}^*))\mathbf{v}^k = \mathbf{v}^k - \alpha\Delta t L(\mathbf{v}^k;\mathbf{u}^*)$$

(2) evaluate the right-hand-side $\mathbf{f}(\mathbf{v})$, and (3) evaluate the linearized operator $L(\mathbf{v};\mathbf{u}^*)$ or the combination $\mathbf{f}(\mathbf{v}) - L(\mathbf{v};\mathbf{u}^*)$.

We could use a predictor step with an second-order accurate explicit predictor for $\mathbf{f}_E$,

$$\frac{\mathbf{u}^p - \mathbf{u}^n}{\Delta t} = \alpha L(\mathbf{u}^{n+1};\mathbf{u}^*) + (1-\alpha)L(\mathbf{u}^n;\mathbf{u}^*) + \frac{3}{2}\mathbf{f}_E(\mathbf{u}^n;\mathbf{u}^*) - \frac{1}{2}\mathbf{f}_E(\mathbf{u}^{n-1};\mathbf{u}^*).$$

With $\alpha = 1/2$ the predicted value $\mathbf{u}^p$ would be second order.

5.4 Example linearizations

Here is an example for the INS:

$$\begin{aligned}
\mathbf{f}(\mathbf{u}) &= \nu\Delta\mathbf{u} - (\mathbf{u}\cdot\nabla)\mathbf{u} - \nabla p \\
L(\mathbf{u};\mathbf{u}^*) &= \nu\Delta\mathbf{u} - \Big((\mathbf{u}^*\cdot\nabla)\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u}^*\Big) \\
\mathbf{f}_E(\mathbf{u};\mathbf{u}^*) &= \mathbf{f}(\mathbf{u}) - L(\mathbf{u};\mathbf{u}^*) \\
&= -(\mathbf{u}\cdot\nabla)\mathbf{u} - \nabla p + \Big((\mathbf{u}^*\cdot\nabla)\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u}^*\Big) \\
&= -\nabla p - ((\mathbf{u}-\mathbf{u}^*)\cdot\nabla)(\mathbf{u}-\mathbf{u}^*) + (\mathbf{u}^*\cdot\nabla)\mathbf{u}^*
\end{aligned}$$

When the viscosity is a function of $\mathbf{u}$, $\nu = \nu(\mathbf{u})$, we could use Here is an example for the INS:

$$\begin{aligned}
\mathbf{f}(\mathbf{u}) &= \nabla\cdot(\nu(\mathbf{u})\nabla\mathbf{u}) - (\mathbf{u}\cdot\nabla)\mathbf{u} - \nabla p \\
L(\mathbf{u};\mathbf{u}^*) &= \nabla\cdot(\nu(\mathbf{u}^*)\nabla\mathbf{u}) + \nabla\cdot(\nu(\mathbf{u})\nabla\mathbf{u}^*) - \Big((\mathbf{u}^*\cdot\nabla)\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u}^*\Big) \\
\mathbf{f}_E(\mathbf{u};\mathbf{u}^*) &= \mathbf{f}(\mathbf{u}) - L(\mathbf{u};\mathbf{u}^*) \\
&= \nabla\cdot(\nu(\mathbf{u})\nabla\mathbf{u}) - \nabla\cdot(\nu(\mathbf{u}^*)\nabla\mathbf{u}) - \nabla\cdot(\nu(\mathbf{u})\nabla\mathbf{u}^*) \\
&\quad - (\mathbf{u}\cdot\nabla)\mathbf{u} - \nabla p + \Big((\mathbf{u}^*\cdot\nabla)\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u}^*\Big) \\
&= \nabla\cdot((\nu(\mathbf{u}) - \nu(\mathbf{u}^*))\nabla(\mathbf{u}-\mathbf{u}^*)) - \nabla\cdot(\nu(\mathbf{u}^*)\nabla\mathbf{u}^*) \\
&\quad - \nabla p - ((\mathbf{u}-\mathbf{u}^*)\cdot\nabla)(\mathbf{u}-\mathbf{u}^*) + (\mathbf{u}^*\cdot\nabla)\mathbf{u}^*
\end{aligned}$$

5.5 Variable time stepping

The `variableTimeStepAdamsPredictorCorrector` time stepping option allows each grid to have it's own time step.

6 Divergence Damping

The divergence damping term, $C_{d,i}\nabla_h \cdot \mathbf{V}_i$, appears in the pressure equation. In simplified terms, the coefficient C_d is taken proportional to the inverse of the time step, $C_d \sim \frac{1}{\Delta t}$. In practice we have found better results by taking $C_d \sim \frac{\nu}{\Delta x^2}$. For explicit time stepping theses are very similar since the explicit time step restriction is something like $\frac{\nu \Delta t}{\Delta x^2} < C$. To allow for the case $\nu = 0$ we use the minumum grid spacing, $h_{\min}$, instead of ν, if $h_{\min} > \nu$. The size of C_d affects the time step, the stability condition is proportional to $C_d \Delta t$. As a result we do not want C_d to be much larger than $1/\Delta t$, and thus it is limited by $\frac{C_t}{\Delta t}$ where C_t is a constant with default value of 0.25. (Note that we don't actually know the true Δt at this point, it depends on C_d, so we just use a guess).

Here is the actual formula for the divergence damping coefficient:

$$C_{d,i} = \min(\mathcal{D}_i, \frac{C_t}{\Delta t})$$

where

$$\begin{aligned}
\mathcal{D}_i &= C_0 \ \max(\nu, h_{\min}) \ \left(\frac{1}{(\Delta_{0,r_1} x_{1,i})^2} + \frac{1}{(\Delta_{0,r_2} x_{2,i})^2} + \frac{1}{(\Delta_{0,r_3} x_{3,i})^2} \right) \\
\Delta_{0,r_1} x_{m,i} &= \frac{1}{2}(x_{m,i_1+1} - x_{m,i_1-1}) \qquad \text{(undivided second difference)} \\
h_{\min} &= \min_i(\|\Delta_{+,r_1}\mathbf{x}_i\|, \Delta_{+,r_2}\mathbf{x}_i\|, \Delta_{+,r_3}\mathbf{x}_i\|) \qquad \text{(minimum grid spacing)}
\end{aligned}$$

and where $C_0 = 1.$ by default.

6.1 Artificial Diffusion

Cgins implements artificial diffusions based either on a second-order undivided difference or a fourth-order undivided difference.

The artificial diffusions are

$$\mathbf{d}_{2,i} = (\texttt{ad21} + \texttt{ad22}|\nabla_h \mathbf{V}_i|_1) \sum_{m=1}^{n_d} \Delta_{m+}\Delta_{m-}\mathbf{V}_i \tag{20}$$

in the second-order case and

$$\mathbf{d}_{4,i} = -(\texttt{ad41} + \texttt{ad42}|\nabla_h \mathbf{V}_i|_1) \sum_{m=1}^{n_d} \Delta_{m+}^2\Delta_{m-}^2\mathbf{V}_i \tag{21}$$

in the fourth-order case. Here $|\nabla_h \mathbf{V}_i|_1$ is the magnitude of the gradient of the velocity and $\Delta_{m\pm}$ are the forward and backward undivided difference operators in direction m

$$\begin{aligned}
|\nabla_h \mathbf{V}_i|_1 &= n_d^{-2} \textstyle\sum_{m=1}^{n_d} \sum_{n=1}^{n_d} |D_{m,h} V_{ni}| \\
\Delta_{1+}\mathbf{V}_i &= \mathbf{V}_{i_1+1} - \mathbf{V}_i \\
\Delta_{1-}\mathbf{V}_i &= \mathbf{V}_i - \mathbf{V}_{i_1+1} \\
\Delta_{2+}\mathbf{V}_i &= \mathbf{V}_{i_2+1} - \mathbf{V}_i \\
\Delta_{2-}\mathbf{V}_i &= \mathbf{V}_i - \mathbf{V}_{i_2+1} \quad \text{etc.}
\end{aligned}$$

The artificial diffusion is added to the momentum equations

$$\frac{d}{dt}\mathbf{V}_i + (\mathbf{V}_i \cdot \nabla_h)\mathbf{V}_i + \nabla_h P_i - \nu\Delta_h \mathbf{V}_i - \mathbf{f}(\mathbf{x}_i, t) - \mathbf{d}_{m,i} = 0$$

but does not change the pressure equation. Typical choices for the constants `ad21` = `ad41` = 1 and `ad22` = `ad42` = 1.. These artificial diffusions should not affect the order of accuracy of the method. With the artificial diffusion turned on to a sufficient degree, the real viscosity can be set at low as zero, `nu` = 0.

In difficult cases you may need to increase the coefficients `ad21` and `ad22` = 1. to keep the solution stable. I suggest trying `ad21` = `ad22` = 2. then 4. etc. until the solution doesn't blow up. I don't think I have ever needed a value larger than 10..

This form of the artificial diffusion is based on a theoretical result [11][12] that states that the minimum scale, $\lambda_{\min}$, of solutions to the incompressible Navier-Stokes equations is proportional to the square root of the kinematic viscosity divided by the square root of the maximum velocity gradient:

$$\lambda_{\min} \propto \sqrt{\frac{\nu}{|\nabla \mathbf{u}| + c}} \; .$$

This result is valid locally in space so that $|\nabla \mathbf{u}|$ measures the local value of the velocity gradient. The minimum scale measures the size of the smallest eddy or width of the sharpest shear layer as a function of the viscosity and the size of the gradients of $\mathbf{u}$. Scales smaller than the minimum scale are in the exponentially small part of the spectrum.

This result can be used to tell us the smallest value that we can choose for the (artificial) viscosity, ν_A, and still obtain a reasonable numerical solution. We require that the artificial viscosity be large enough so that the smallest (but still significant) features of the flow are resolved on the given mesh. If the local grid spacing is h, then we need

$$h \propto \sqrt{\frac{\nu_A}{|\nabla \mathbf{u}| + c}} \; .$$

This gives

$$\nu_A = (c_1 + c_2 |\nabla \mathbf{u}|) h^2$$

and thus we can choose an artificial diffusion of

$$(c_1 + c_2 |\nabla \mathbf{u}|) h^2 \Delta \mathbf{u}$$

which is just the form (20).

In the fourth-order accurate case we wish to add an artificial diffusion of the form

$$-\nu_A \Delta^2 \mathbf{u}$$

since, as we will see, this will lead to $\nu_A \propto h^4$. In this case, if we consider solutions to the incompressible Navier-Stokes equations with the diffusion term $\nu \Delta \mathbf{u}$ replaced by $-\nu_A \Delta^2 \mathbf{u}$ then the minimum scale would be

$$\lambda_{\min} \propto \left(\frac{\nu_A}{|\nabla \mathbf{u}|} \right)^{1/4}$$

Following the previous argument leads us to choose an artificial diffusion of the form

$$-(c_1 + c_2 |\nabla \mathbf{u}|) h^4 \Delta^2 \mathbf{u}$$

which is just like (21).

7 Boundary Conditions

The boundary conditions for method INS are

$$\text{noSlipWall} = \begin{cases} \mathbf{u} = \mathbf{g} & \text{velocity specified} \\ \nabla \cdot \mathbf{u} = 0 & \text{divergence zero} \end{cases}$$

$$\text{slipWall} = \begin{cases} \mathbf{n} \cdot \mathbf{u} = g & \text{normal velocity specified} \\ \partial_n(\mathbf{t}_m \cdot \mathbf{u}) = 0 & \text{normal derivative of tangential velcity is zero} \\ \nabla \cdot \mathbf{u} = 0 & \text{divergence zero} \end{cases}$$

$$\text{inflowWithVelocityGiven} = \begin{cases} \mathbf{u} = \mathbf{g} & \text{velocity specified} \\ \partial_n p = 0 & \text{normal derivative of the pressure zero.} \end{cases}$$

$$\text{outflow} = \begin{cases} \text{extrapolate } \mathbf{u} & \\ \alpha p + \beta \partial_n p = g & \text{mixed derivative of p given.} \end{cases}$$

$$\text{symmetry} = \begin{cases} \mathbf{n} \cdot \mathbf{u}\text{: odd}, \mathbf{t}_m \cdot \mathbf{u}\text{: even} & \text{vector symmetry} \\ \partial_n p = 0 & \text{normal derivative of the pressure zero.} \end{cases}$$

$$\text{dirichletBoundaryCondition} = \begin{cases} \mathbf{u} = \mathbf{g} & \text{velocity specified} \\ p = P & \text{pressure given} \end{cases}$$

8 Boundary conditions for the fourth-order method

Here are the analytic and numerical conditions that we impose at a boundary inorder to determine the values of $\mathbf{u}$ at the two ghost points.

noSlipWall: Analytic boundary conditions

$$\mathbf{u} = \mathbf{u}_B(\mathbf{x}, t)$$

plus numerical boundary conditions

$$\begin{aligned} \mathbf{t}_\mu \cdot \Big\{ \nu \Delta \mathbf{u} - \nabla p - (\mathbf{u} \cdot \nabla)\mathbf{u} - \mathbf{u}_t \Big\} &= 0 \\ \text{Extrapolate} \quad \mathbf{t}_\mu \cdot \mathbf{u} &= 0 \\ \nabla \cdot \mathbf{u} &= 0 \\ \partial_n (\nabla \cdot \mathbf{u}) &= 0 \end{aligned}$$

inflowWithVelocityGiven or outflow: Analytic boundary conditions for inflow are

$$\mathbf{u} = \mathbf{u}_I(\mathbf{x}, t) \qquad \text{(inflow)}$$

For outflow the equation is used on the boundary. The numerical boundary conditions are

$$\begin{aligned} \mathbf{t}_\mu \cdot (\mathbf{u}_{nn}) &= 0 \\ \text{Extrapolate} \quad \mathbf{t}_\mu \cdot \mathbf{u} &= 0 \\ \nabla \cdot \mathbf{u} &= 0 \\ \partial_n (\nabla \cdot \mathbf{u}) &= 0 \end{aligned}$$

slipWall: Analytic boundary conditions are

$$\mathbf{n} \cdot \mathbf{u} = \mathbf{n} \cdot \mathbf{u}_B$$

The numerical boundary conditions are

$$\begin{aligned} \mathbf{t}_\mu \cdot \Big\{ \nu \Delta \mathbf{u} - \nabla p - (\mathbf{u} \cdot \nabla)\mathbf{u} - \mathbf{u}_t \Big\} &= 0 \quad \text{determines } \mathbf{t}_\mu \cdot \mathbf{u} \text{ on the boundary} \\ \mathbf{t}_\mu \cdot (\mathbf{u}_n) &= 0 \\ \mathbf{t}_\mu \cdot (\mathbf{u}_{nnn}) &= 0 \\ \nabla \cdot \mathbf{u} &= 0 \\ \partial_n (\nabla \cdot \mathbf{u}) &= 0 \end{aligned}$$

Discretizing the Boundary conditions: For the purposes of this discussion assume that the boundary condition for $\mathbf{u}$ is of the form $\mathbf{u}(\mathbf{x}, t) = \mathbf{u}_B(\mathbf{x}, t)$ for $\mathbf{x} \in \partial\Omega$. More general boundary conditions on $\mathbf{u}$ and p, such as extrapolation conditions, can also be dealt with although some of the details of implementation may vary. At a boundary the following conditions are applied

$$\left.\begin{aligned} \mathbf{U}_i - \mathbf{u}_B(\mathbf{x}_i) &= 0 \\ \nabla_4 \cdot \mathbf{U}_i &= 0 \\ D_{4n}(\nabla_4 \cdot \mathbf{U}_i) &= 0 \\ \tfrac{d}{dt}\mathbf{U}_i + (\mathbf{U}_i \cdot \nabla_4)\mathbf{U}_i + \nabla_4 P_i - \nu \Delta_4 \mathbf{U}_i - \mathbf{f}_i &= 0 \\ \Delta_4 P_i + \sum_{m=1}^{n_d} \nabla_4 U_{m,i} \cdot D_{4x_m} \mathbf{U}_i - \nabla_4 \cdot \mathbf{f}_i &= 0 \end{aligned}\right\} \quad \text{for } i \in \text{Boundary}$$

$$\left.\begin{aligned} \mathbf{t}_\mu \cdot D^4_{+m} \mathbf{U}_i &= 0 \\ D^4_{+m} P_i &= 0 \end{aligned}\right\} \quad \text{for } i \in \text{2nd fictitious line}$$

where $\mathbf{t}_\mu$, $\mu = 1, n_d - 1$ are linearly independent vectors that are tangent to the boundary. In the extrapolation conditions either D_{+m} or D_{-m} should be chosen, as appropriate. Thus at each point along the boundary there are 12 equations for the 12 unknowns $(\mathbf{U}_i, P_i)$ located on the boundary and the 2 lines of fictitious points. Note that two of the numerical boundary conditions couple the pressure and velocity. In order to advance the velocity with an explicit time stepping method it convenient to decouple the solution of the pressure equation from the solution of the velocity. A procedure to accomplish this is described in the next section on time stepping.

Edges and Vertices: An important special case concerns obtaining solution values at points that lie near edges and vertices of grids (or corners of grids in 2D). Define a *boundary edge* to be the edge that is formed at the intersection of adjacent faces of the unit cube where both faces are boundaries of the computational domain. Along a boundary edge, values of the solution are required at the fictitious points in the region exterior to both boundary faces. For example, suppose that the edge defined by $i_1 = n_{1,a}$, $i_2 = n_{2,a}$ and $i_3 = n_{3,a}, \ldots, n_{3,b}$ is a boundary edge. Values must be determined at the exterior points $i = (n_{1,a} + m, n_{2,a} + n, i_3)$ for $m, n = -2, -1$.

Here we derive a more accurate formula than was in my paper. These expressions will be exact for polynomials of degree 4. By Taylor series,

$$u(r_1, r_2) = u(0,0) + \mathcal{D}_1(r_1, r_2) + \mathcal{D}_2(r_1, r_2) + \mathcal{D}_3(r_1, r_2) + \mathcal{D}_4(r_1, r_2) + O(|\mathbf{r}|^6)$$

where

$$\begin{aligned}
\mathcal{D}_1(r_1, r_2) &= (r_1 \partial_{r_1} + r_2 \partial_{r_2}) u(0,0) \\
\mathcal{D}_2(r_1, r_2) &= \frac{1}{2}({r_1}^2 \partial^2_{r_1} + {r_2}^2 \partial^2_{r_2} + 2 r_1 r_2 \partial_{r_1} \partial_{r_2}) u(0,0) \\
\mathcal{D}_3(r_1, r_2) &= \frac{1}{3!}({r_1}^3 \partial^3_{r} + {r_2}^3 \partial^3_{r_2} + 3{r_1}^2 r_2 \partial^2_{r_1} \partial_{r_2} + 3 r_1 {r_2}^2 \partial_{r_1} \partial^2_{r_2}) u(0,0)
\end{aligned}$$

We also have

$$u(-r_1, -r_2) = 2u(0,0) - u(r_1, r_2) + 2\mathcal{D}_2(r_1, r_2) + 2\mathcal{D}_4(r_1, r_2) + O(|\mathbf{r}|^6) \tag{22}$$

$$u(2r_1, 2r_2) = u(0,0) + 2\mathcal{D}_1(r_1, r_2) + 4\mathcal{D}_2(r_1, r_2) + 8\mathcal{D}_3(r_1, r_2) + 16\mathcal{D}_4(r_1, r_2) + O(|\mathbf{r}|^6) \tag{23}$$

$$8u(r_1, r_2) - u(2r_1, 2r_2) = 7u(0,0) + 6\mathcal{D}_1(r_1, r_2) + 4\mathcal{D}_2(r_1, r_2) - 8\mathcal{D}_4(r_1, r_2) + O(|\mathbf{r}|^6) \tag{24}$$

From equation (24) we can solve for $\mathcal{D}_4(r_1, r_2)$,

$$\mathcal{D}_4(r_1, r_2) = \frac{7}{8}u(0,0) - u(r_1, r_2) + \frac{1}{8}u(2r_1, 2r_2) + \frac{3}{4}\mathcal{D}_1(r_1, r_2) + \frac{1}{2}\mathcal{D}_2(r_1, r_2) + O(|r|^5 + |s|^5)$$

and substitute into equation (22)

$$u(-r_1, -r_2) = \frac{15}{4}u(0,0) - 3u(r_1, r_2) + \frac{1}{4}u(2r_1, 2r_2) + \frac{3}{2}\mathcal{D}_1(r_1, r_2) + 3\mathcal{D}_2(r_1, r_2) + O(|\mathbf{r}|^6) \tag{25}$$

We will use this last equation to determine u at the first corner ghost point, $\mathbf{U}_{-1,-1,i_3}$. Proceeding in a similar way it follows that

$$u(-2r_1, -r_2) = \frac{15}{4}u(0,0) - 3u(2r_1, r_2) + \frac{1}{4}u(4r_1, 2r_2) + \frac{3}{2}\mathcal{D}_1(2r_1, r_2) + 3\mathcal{D}_2(2r_1, r_2) + O(|\mathbf{r}|^6)O(|\mathbf{r}|^6) \tag{26}$$

$$u(-r_1, -2r_2) = \frac{15}{4}u(0,0) - 3u(r_1, 2r_2) + \frac{1}{4}u(2r_1, 4r_2) + \frac{3}{2}\mathcal{D}_1(r_1, 2r_2) + 3\mathcal{D}_2(r_1, 2r_2) + O(|\mathbf{r}|^6)O(|\mathbf{r}|^6) \tag{27}$$

$$u(-2r_1, -2r_2) = 30u(0,0) - 32u(r_1, r_2) + 3u(2r_1, 2r_2) + 24\mathcal{D}_1(r_1, r_2) + 24\mathcal{D}_2(r_1, r_2) + O(|\mathbf{r}|^6)O(|\mathbf{r}|^6) \tag{28}$$

from which we will determine the ghost points values at $\mathbf{U}_{-2,-2,i_3}$, $\mathbf{U}_{-2,-1,i_3}$ and $\mathbf{U}_{-1,-2,i_3}$. By symmetry we obtain formulae for ghost points outside all other edges in three-dimensions, $\mathbf{U}_{-2:-1,i_2,-2:-1}$, $\mathbf{U}_{i_1,-2:-1,-2:-1}$. For ghost points outside the vertices in three-dimensions we have

$$u(-r_1,-r_2,-r_3) = \frac{15}{4}u(0,0) - 3u(r_1,r_2,r_3) + \frac{1}{4}u(2r_1,2r_2,2r_3) \tag{29}$$

$$+ \frac{3}{2}\mathcal{D}_1(r_1,r_2,r_3) + 3\mathcal{D}_2(r_1,r_2,r_3) + O(|\mathbf{r}|^6)O(|\mathbf{r}|^6) \tag{30}$$

where

$$\mathcal{D}_1(r_1,r_2,r_3) = \Big(r_1\partial_{r_1} + r_2\partial_{r_2} + r_3\partial_{r_3}\Big)u(0,0,0)$$

$$\mathcal{D}_2(r_1,r_2,r_3) = \frac{1}{2}\Big({r_1}^2\partial^2_{r_1} + {r_2}^2\partial^2_{r_2} + {r_3}^2\partial^2_{r_3} + 2r_1r_2\partial_{r_1}\partial_{r_2} + 2r_1r_3\partial_{r_1}\partial_{r_3} + 2r_2r_3\partial_{r_2}\partial_{r_3}\Big)u(0,0,0)$$

$$\mathcal{D}_3(r_1,r_2,r_3) = \frac{1}{3!}\sum_{m_1=1}^{3}\sum_{m_2=1}^{3}\sum_{m_3=1}^{3} r_{m_1}r_{m_2}r_{m_3}\partial_{m_1}\partial_{m_2}\partial_{m_3}u(0,0,0)$$

$$= \frac{1}{3!}\Big({r_1}^3\partial^3_r + {r_2}^3\partial^3_{r_2} + {r_3}^3\partial^3_{r_3} + 3{r_1}^2r_2\partial^2_{r_1}\partial_{r_2} + 3r_1{r_2}^2\partial_{r_1}\partial^2_{r_2}$$

$$+ 3{r_1}^2r_3\partial^2_{r_1}\partial_{r_3} + 3r_1{r_3}^2\partial_{r_1}\partial^2_{r_3} + 3{r_2}^2r_3\partial^2_{r_2}\partial_{r_3} + 3r_2{r_3}^2\partial_{r_2}\partial^2_{r_3} + 6r_1r_2r_3\partial_{r_1}\partial_{r_2}\partial_{r_3}\Big)u(0,0,0)$$

In order to evaluate the formulae (25,26,27, 28,30) we need to evaluate the derivatives appearing in $\mathcal{D}_1$ and $\mathcal{D}_2$. All the non-mixed derivatives $\partial^m u(0,0)/\partial^m_{r_n}$, $m = 1, 2$, can be evaluated using the boundary values since these are all tangential derivatives. The second-order mixed derivative term such as $u_{r_1r_2}$ requires a bit more work. In two-dimensions we evaluate this term by taking the parametric derivatives of the divergence, $\partial_{r_m}\nabla\cdot\mathbf{u} = 0$. Since

$$\nabla\cdot\mathbf{u} = \sum_{n=1}^{2}(\partial_x r_n)\ u_{r_n} \ + \ \sum_{n=1}^{2}(\partial_y r_n)\ v_{r_n}$$

then $\partial_{r_m}\nabla\cdot\mathbf{u} = 0$ gives

$$\begin{aligned}(\partial_x r_2)\ u_{r_1r_2} + (\partial_y r_2)\ v_{r_1r_2} =& (\partial_x r_2)_{r_1}\ u_{r_2} + (\partial_x r_1)\ u_{r_1r_1} + (\partial_x r_1)_{r_1}\ \partial_{r_1}u + \\ & (\partial_y r_2)_{r_1}\ v_{r_2} + (\partial_y r_1)\ v_{r_1r_1} + (\partial_y r_1)_{r_1}\ \partial_{r_1}v \\ (\partial_x r_1)\ u_{r_1r_2} + (\partial_y r_1)\ v_{r_1r_2} =& (\partial_x r_2)\ u_{r_2r_2} + (\partial_x r_2)_{r_2}\ \partial_{r_2}u ...\end{aligned}$$

These last two equations can be solved for $u_{r_1r_2}$ and $v_{r_1r_2}$ in terms of known tangential derivatives.

In three dimensions taking the two parametric derivatives of the divergence,

$$(\partial_x r_2)\ u_{r_1r_2} + (\partial_y r_2)\ v_{r_1r_2} + (\partial_z r_2)\ w_{r_1r_2} = ... \tag{31}$$

$$(\partial_x r_1)\ u_{r_1r_2} + (\partial_y r_1)\ v_{r_1r_2} + (\partial_z r_1)\ w_{r_1r_2} = ... \tag{32}$$

gives only two equations for the three unknowns, $u_{r_1r_2}, v_{r_1r_2}$ and $w_{r_1r_2}$. We therefore add an extra condition by extrapolating the tangential component of the velocity,

$$\mathbf{t}_3 \cdot D^6_{+,1,2}\mathbf{U}_{i_1-1,i_2-1,i_3} = 0 \tag{33}$$

Solve the last equation for $\mathbf{t}_3\cdot\mathbf{U}_{i_1-1,i_2-1,i_3}$ gives

$$\mathbf{t}_3\cdot\mathbf{U}_{i_1-1,i_2-1,i_3} = \mathcal{E}^6_{+,1,2}\mathbf{t}_3\cdot\mathbf{U}_{i_1-1,i_2-1,i_3} \tag{34}$$

where we have introduced the operator $\mathcal{E}^6_{+,1,2}$. By substituting this last equation (34) into $\mathbf{t}_3\cdot$ equation (25),

$$\mathbf{t}_3\cdot\mathbf{u}(-r_1,-r_2) = \mathbf{t}_3\cdot\left(\frac{15}{4}\mathbf{u}(0,0) - 3\mathbf{u}(r_1,r_2) + \frac{1}{4}\mathbf{u}(2r_1,2r_2) + \frac{3}{2}\mathcal{D}_1(r_1,r_2)\mathbf{u}(0) + 3\mathcal{D}_2(r_1,r_2)\mathbf{u}(0)\right) + O(|\mathbf{r}|^6)$$

we can eliminate $\mathbf{t}_3 \cdot \mathbf{U}_{i_1-1,i_2-1,i_3}$ and obtain an equation for $\mathbf{t}_3 \cdot \mathbf{u}_{r_1 r_2}$ in terms of known quanitites,

$$\mathbf{t}_3 \cdot \left(\mathcal{E}^6_{+,1,2}\mathbf{U}_{i_1-1,i_2-1,i_3}\right) = \mathbf{t}_3 \cdot \Big(\frac{15}{4}\mathbf{u}(0,0) - 3\mathbf{u}(r_1,r_2) + \frac{1}{4}\mathbf{u}(2r_1,2r_2) + \frac{3}{2}\mathcal{D}_1(r_1,r_2)\mathbf{u}(0) + \frac{3}{2}({r_1}^2\partial^2_{r_1} + {r_2}^2\partial^2_{r_2} + {r_3}^2\partial^2_{r_3} + 2r_1r_2\partial_{r_1}\partial_{r_2} + 2r_1r_3\partial_{r_1}\partial_{r_3} + 2r_2r_3\partial_{r_2}\partial_{r_3})\mathbf{u}(0,0)\Big)$$

or re-written as

$$\mathbf{t}_3 \cdot \mathbf{u}_{r_1 r_2}(0,0) = \frac{1}{3}\Big\{\mathbf{t}_3 \cdot \left(\mathcal{E}^6_{+,1,2}\mathbf{U}_{i_1-1,i_2-1,i_3}\right) - \mathbf{t}_3 \cdot \Big(\frac{15}{4}\mathbf{u}(0,0) - 3\mathbf{u}(r_1,r_2) + \frac{1}{4}\mathbf{u}(2r_1,2r_2) + \frac{3}{2}\mathcal{D}_1(r_1,r_2)\mathbf{u}(0) + \frac{3}{2}({r_1}^2\partial^2_{r_1} + {r_2}^2\partial^2_{r_2} + {r_3}^2\partial^2_{r_3} + 2r_1r_3\partial_{r_1}\partial_{r_3} + 2r_2r_3\partial_{r_2}\partial_{r_3})\mathbf{u}(0,0)\Big\} \tag{35}$$

To summarize we solve equations (31,32,35) for the three unknowns $u_{r_1r_2}, v_{r_1r_2}$ and $w_{r_1r_2}$.

Solving the numerical boundary equations: The numerical boundary conditions (**??**) define the values of $\mathbf{U}$ on two lines of fictitious points in terms of values of the velocity on the boundary and the interior. The equations couple the unknowns in the tangential direction to the boundary so that in principle a system of equations for all boundary points must be solved. However, when the grid is nearly orthogonal to the boundary there is a much more efficient way to solve the boundary conditions. The first step in the algorithm is to solve for the tangential components of the velocity from

$$\left.\begin{aligned} \mathbf{U}_i(t) - \mathbf{u}_B(\mathbf{x}_i,t) &= 0 \\ \mathbf{t}_\mu \cdot \Big\{\tfrac{d}{dt}\mathbf{U}_i(t) + (\mathbf{U}_i(t)\cdot\nabla_4)\mathbf{U}_i(t) + \nabla_4 P^*(t) - \nu\Delta_4\mathbf{U}_i(t) - \mathbf{f}\Big\} &= 0 \end{aligned}\right\} \quad \text{for } i \in \text{Boundary}$$

$$\left.\begin{aligned} \mathbf{t}_\mu \cdot D^6_{+m}(\mathbf{U}_i(t)) &= 0 \end{aligned}\right\} \quad \text{for } i \in \text{Second fictitious line}$$

If the grid is orthogonal to the boundary then the discrete Laplacian applied at boundary will not have any mixed derivative terms. Therefore the only fictitious points appearing in the equation applied at the the boundary point (i_1,i_2,i_3) will be the two points (i_1,i_2,i_3-n) $n=1,2$ (here we assume that i_3 is in the normal direction to the boundary). Thus for each point on the boundary (i_1,i_2,i_3) the values of $\mathbf{t}_\mu \cdot \mathbf{u}$ can be determined at the fictitious points (i_1,i_2,i_3-1) and (i_1,i_2,i_3-2). There is no coupling between adjacent boundary points so no large system of equations need be solved. The tangential components of the velocity are determined for all fictitious points on the entire boundary. The second step is to determine the the normal component of the velocity at the fictitious points from

$$\left.\begin{aligned} \mathbf{U}_i(t) - \mathbf{u}_B(\mathbf{x}_i,t) &= 0 \\ \nabla_4 \cdot \mathbf{U}_i(t) &= 0 \\ D_{4n}(\nabla_4 \cdot \mathbf{U}_i(t)) &= 0 \end{aligned}\right\} \quad \text{for } i \in \text{Boundary}$$

If the grid is orthogonal to the boundary then the divergence on the boundary can be written in the form

$$\nabla \cdot \mathbf{u} = \frac{1}{e_1e_2e_3}\left\{\frac{\partial}{\partial n}(e_2e_3\mathbf{n}\cdot\mathbf{u}) + \frac{\partial}{\partial t_1}(e_1e_3\mathbf{t}_1\cdot\mathbf{u}) + \frac{\partial}{\partial t_2}(e_1e_2\mathbf{t}_2\cdot\mathbf{u})\right\}$$

where the e_m are functions of $\partial\mathbf{x}/\partial\mathbf{r}$. Note that only normal derivatives of $\mathbf{n}\cdot\mathbf{u}$ appear in the expression for the divergence. Thus, at a boundary point, (i_1,i_2,i_3), the stencil for $\nabla_4\cdot\mathbf{U}$ will only involve the fictitious points at (i_1,i_2,i_3-n), $n=1,2$. Similarly, the stencil for $D_{4n}(\nabla_4\cdot\mathbf{U})$ at a boundary will only involve the fictitious points at (i_1,i_2,i_3-n), $n=1,2$. Thus there is no coupling between adjacent boundary points and the unknown values for $\mathbf{n}\cdot\mathbf{u}$ can be easily determined. Note that the equations for $D_{4n}(\nabla_4\cdot\mathbf{U})$ will couple values for $\mathbf{t}_\mu\cdot\mathbf{u}$ at fictitious points along the boundary but these values have already been determined in the first step.

In practice the boundary conditions are solved in a correction mode – some initial guess is assumed for the values at the fictitious points and a correction is computed. If the grid is orthogonal or nearly orthogonal to the boundary then the first correction will give an accurate answer to the boundary conditions. If the grid is not orthogonal to the boundary then the solution procedure can repeated one or more times until a desired accuracy is achieved. This iteration should converge quickly provided that the grid is not overly skewed.

9 Turbulence models

The typical RANS model for the incompressible Navier-Stokes equations which uses the Boussinesq eddy viscosity approximation is

$$\begin{aligned}\partial_t u_i + u_k \partial_{x_k} u_i + \partial_{x_i} p &= \partial_{x_k}\Big((\nu + \nu_T)(\partial_{x_k} u_i + \partial_{x_i} u_k)\Big)\\ \tau_{i,j} &= (\nu + \nu_T)(\partial_{x_k} u_i + \partial_{x_i} u_k)\\ \widetilde{\nu} &= \nu + \nu_T\end{aligned}$$

where ν_T is the turbulent eddy viscosity.

9.1 Wall Shear Stress

The stress on a wall with normal $\mathbf{n}$ is

$$\boldsymbol{\tau}_{\text{wall}} = \boldsymbol{\tau} \cdot \mathbf{n}$$

The shear stress is the tangential component of this wall stress,

$$\begin{aligned}\boldsymbol{\tau}_{\text{shear}} &= \boldsymbol{\tau}_{\text{wall}} - (\mathbf{n} \cdot \boldsymbol{\tau}_{\text{wall}})\mathbf{n}\\ &= \tau_{i,j} n_j - (n_k \tau_{k,j} n_j) n_i\end{aligned}$$

In two-dimensions for a horizontal wall, $\mathbf{n} = [0, 1]^T$ and the wall shear stress is

$$\tau_w = (\nu + \nu_T)(\partial u/\partial y + \partial v/\partial x)$$

Question : Is ν_T on the wall equal to zero always?

In three-dimensions we define scalar wall shear stress τ_w in terms of the tangential component of the velocity at the wall, $\mathbf{u}_t$,

$$\begin{aligned}\tau_w &= \mathbf{t} \cdot \boldsymbol{\tau} \cdot \mathbf{n},\\ \mathbf{u}_{\tan} &= \mathbf{u} - (\mathbf{n} \cdot \mathbf{u})\mathbf{n},\\ \mathbf{t} &= \mathbf{u}_{\tan}/|\mathbf{u}_{\tan}|.\end{aligned}$$

Given the definition τ_w we can define (see for example Wilcox [14])

$$\begin{aligned}u_\tau &= \sqrt{\frac{\tau_w}{\rho}} && \text{Friction velocity}\\ \mathbf{u}^+ &= \mathbf{u}/u_\tau && \text{non-dimensional velocity for near wall region}\\ y^+ &= u_\tau y/\nu && \text{non-dimensional length for near wall region}\\ U &= |\mathbf{u}_{\tan}|\\ \frac{U}{u_\tau} &= \frac{1}{\kappa}\ln\frac{u_\tau y}{\nu} + C && \text{law of the wall, } \kappa\text{=Kármán's constant}\\ U^+ &= \frac{U}{u_\tau} && \text{non-dimensional velocity for near wall region}\\ U^+ &= \frac{1}{\kappa}\ln(Ey^+) && \text{law of the wall, } E\text{=surface roughness parameter}\end{aligned}$$

For a horizontal wall, with flow in the x-direction, and $\rho = 1$, and assuming $\nu_T(0) = 0$, we have

$$\begin{aligned}u_\tau &= \sqrt{\nu U_y(0)}\\ y^+ &= y\sqrt{U_y(0)/\nu}\\ U(y) &= \frac{\sqrt{\nu U_y(0)}}{\kappa}\ln\left[Ey\sqrt{U_y(0)/\nu}\right]\end{aligned}$$

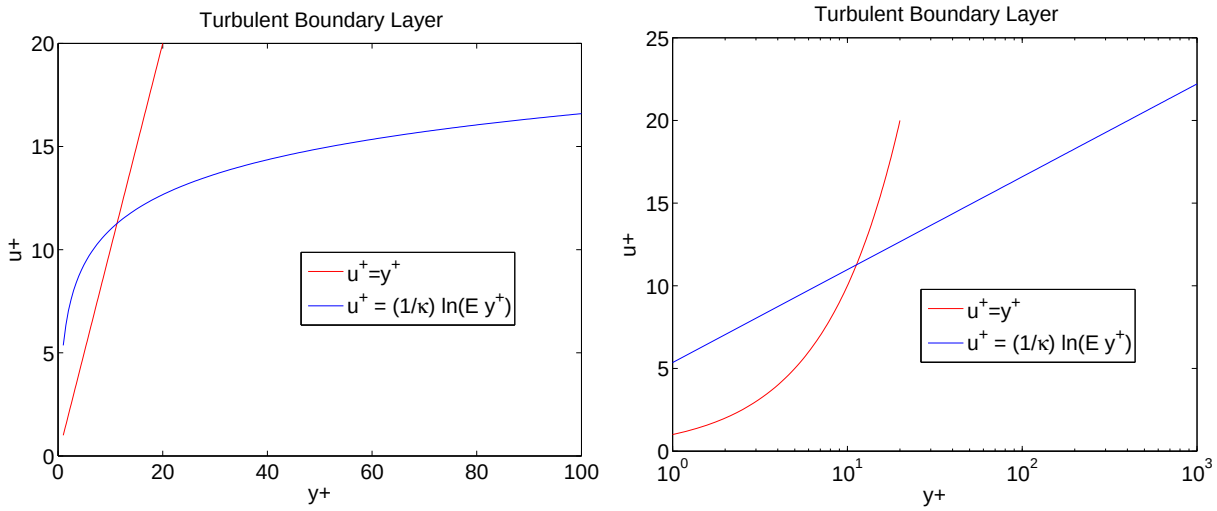


Figure 2: A turbulent boundary layer is thought to have an inner viscous sublayer where $u^+ = y+$ and an outer log-layer where $u^+ = \kappa^{-1}\ln(Ey^+)$.

9.2 Wall Functions

Wall functions are used to apply approximate boundary conditions in numerical simulations. They are used when the grid does not resolve the boundary layer.

The basic wall function approach assumes that the first grid point off the wall is located in the region where the law of the wall is valid. If y_p denotes the first grid point next to the wall then the law of the wall relates the tangential component of the velocity in the direction of the flow, $U(y_p)$, to the wall shear stress,

$$U(x, y_p) = \frac{u_\tau}{\kappa}\ln\left[Ey_p u_\tau/\nu\right] \tag{36}$$

$$U^+ = \kappa^{-1}\ln(Ey^+) \tag{37}$$

$$u_\tau = \sqrt{\nu u_y(x,0)} = \sqrt{\tau_w/\rho} \quad \text{(friction velocity)} \tag{38}$$

Here $E = 9$ for smooth walls and $\kappa = 0.41$. The law of the wall is thought to be valid in an attached turbulent boundary layer in region $\Omega_{\text{wall-law}} = \{\ \mathbf{x}\ |20 \le y^+ \le 100\}$. This condition replaces the wall no-slip boundary condition, $U(0) = 0$, on the component of the velocity in the direction of the flow (i.e. $U = \mathbf{t}\cdot\mathbf{U}$ where $\mathbf{t} = \mathbf{U}/|\mathbf{U}|$) . The condition is combined with the BC $\mathbf{n}\cdot\mathbf{U}(0) = 0$. In 3D we also need a condition on the other tangential component, which I think that we set to zero.

Note that $\lim_{y\to 0} U_y(x,y) \neq u_y(x,0)$ since the law of the wall does not hold for $y^+ < 20$. Thus the wall shear stress is defined using the fully resolved solution $u(x,y)$ and not the Reynold's averaged solution $U(x,y)$.

In the literature it is usually stated that the law of the wall is used to give τ_w, given a value $U(x,y_p)$, and this τ_w is used in the momentum equations to evaluate $\nabla\cdot(\boldsymbol{\tau})$ at the point y_p. This doesn't seem correct. The correct way would be to use the law of the way to evaluate $U_y(x,y)$ for a point near the wall and use this value in the momentum equation.

By differentiating the law of the wall w.r.t to y we get

$$U_y(x,y) = \frac{u_\tau}{\kappa y}\ ,\ U_{yy}(x,y) = -\frac{u_\tau}{\kappa y^2}, \tag{39}$$

which are valid for $\mathbf{x}\in\Omega_{\text{wall-law}}$.

So maybe we can implement the wall function by solving the full momentum equations on the first line in and by defining $U(x,0)$ from the two equations

$$\frac{U(x,y_p)-U(x,0)}{\Delta y} \approx U_y(x,y_p/2) \approx \frac{u_\tau}{\kappa}\frac{1}{y_p/2} \tag{40}$$

$$U(x,y_p) = \frac{u_\tau}{\kappa}\ln\left[Ey_p u_\tau/\nu\right] \tag{41}$$

The second equation gives u_τ as an implicit function of $U(x,y_p)$, $u_\tau = F(U(x,y_p))$

$$U(x,y_p) = \frac{u_\tau}{\kappa}\ln(Ey_p u_\tau/\nu) \leftrightarrow u_\tau = F(U(x,y_p)) \tag{42}$$

$$\tag{43}$$

These equations can be thus used to eliminate u_τ to give $U(x,0)$ in terms of $U(x,y_p)$, giving

$$U(x,0) = U(x,y_p) - \frac{u_\tau}{\kappa}\frac{\Delta y}{y_p/2} \tag{44}$$

$$= U(x,y_p) - \frac{2}{\kappa}F(U(x,y_p)) \tag{45}$$

We can approximate this non-linear boundary condition by linearizing about some current guess for $U(x,y_p) \approx U_p^0$. Suppose that $F(U_p^0) = u_\tau^0$. Then expanding $F(U_p^0+\delta U)$ gives

$$F(U_p^0+\delta U) \approx F(U_p^0) + \frac{\partial F}{\partial U}(U_p^0)\delta U$$

Differentiating the expression (42) with respect to U gives (*check this*)

$$\frac{\partial F}{\partial U} = \frac{\partial u_\tau}{\partial U} = \frac{\kappa}{1+\ln(Ey_p u_\tau/\nu)} = \frac{\kappa}{1+\kappa U/u_\tau}$$

Whence the linearized boundary condition is

$$\begin{aligned}
U(x,0) &= U(x,y_p) - 2\kappa^{-1}F(U(x,y_p)) \\
&= U(x,y_p) - 2\kappa^{-1}\Big(F(U_p^0) + F_U(U_p^0)(U(x,y_p)-U_p^0)\Big) \\
&= \Big(1-2\kappa^{-1}F_U(U_p^0)\Big)U(x,y_p) - 2\kappa^{-1}\Big(F(U_p^0) - F_U(U_p^0)\ U_p^0\Big) \\
&= A_w U(x,y_p) + B_w \\
A_w(U_p^0) &= 1 - \frac{2}{1+\kappa U_p^0/u_\tau^0} = \frac{\kappa U_p^0/u_\tau^0 - 1}{\kappa U_p^0/u_\tau^0 + 1} = \frac{\ln(Ey_p^+)-1}{\ln(Ey_p^+)+1} \\
B_w(U_p^0) &= -2\kappa^{-1}u_\tau^0 + \frac{2}{1+\kappa U_p^0/u_\tau^0}\ U_p^0 = \frac{-2\kappa^{-1}u_\tau^0}{\kappa U_p^0/u_\tau^0+1}
\end{aligned}$$

This gives a boundary condition involving $U(x,0)$ and $U(x,y_p)$ linearized about a state (U_p^0, u_τ^0) that satisfies the law of the wall. Also note that $0 < A_w < 1$ if $Ey_p^+ > e$ (recall that $E = 9$ for smooth walls).

From Wilcox [14], the boundary conditions for k and ϵ for the $k-\epsilon$ model are then given by

$$k = \frac{u_\tau^2}{\sqrt{C_\mu}}, \qquad \epsilon = \frac{u_\tau^3}{\kappa y}$$

These could be imposed on the first line in, $y = y_p$?

Solving for u_τ from the law of the wall: Given a value $U_p = U(x, y_p)$ we need to be able to solve the nonlinear equation (42) to compute a value for $z = u_\tau$ satisfying

$$\begin{aligned} G(z) &\equiv \kappa^{-1} z \ln(Az) - U_p = 0 \\ A &= Ey_p/\nu \\ G'(z) &= \kappa^{-1}(1 + \ln(Az)) \end{aligned}$$

Note: If $U_p = 0$ then $z = 1/A$. Using Newton's method we get the iteration,

$$z^{k+1} = z^k - \frac{G(z^k)}{G'(z^k)}$$

There could be trouble with this iteration if $G'(z^k) = 0$, that is when $Az^k = e^{-1}$. However, if $U_p \geq 0$ then Az should satisfy $Az \geq 1 > e^{-1}$. This can seen from the fact that $z \ln(Az) \leq 0$ for $Az \leq 1$. Therefore $G(z) = 0$ at $z = 1/A$ and thus z should always satisfy $z > 1/A$ which imples $G'(z) \geq \kappa^{-1}$.

An initial guess for z may come from the expression

$$z = \frac{\kappa U_p}{\ln(Ey_p^+)}, \quad \rightarrow \quad \frac{\kappa U_p}{\ln(100E)} < z^0 < \frac{\kappa U_p}{\ln(20E)}$$

if we assume that $y_p^+ \in [20, 100]$.

In a boundary layer we might expect that as $R_e \to \infty$,

$$\begin{aligned} \tau_w &\sim R_e^{-1/2} \ll 1 \\ u_\tau &\sim R_e^{-1/4} \ll 1 \end{aligned}$$

The viscous length scale is $\lambda_{\min} = \nu/u_\tau$. If the first grid line is at $y_p = M\lambda_{\min}$ (i.e. at $y^+ = M$) with $M \in [20, 100]$, then

$$\begin{aligned} A = Ey_p/\nu = EM/u_\tau \sim EMR_e^{1/2} \gg 1 \\ Au_\tau = EM \end{aligned}$$

The product $EM \approx 9M$ is quite large.

Figure 3 shows a plot of U_p verses u_τ for the law-of-the wall function We see that a linear fit $u_\tau = aU_p + 1/A$ is a good approximation where the constant a is chosen as

$$a = \frac{u_\tau^m - 1/A}{\kappa^{-1} u_\tau^m \ln(Au_\tau^m)}$$

so that $U_p = \kappa^{-1} u_\tau^m \ln(Au_\tau^m)$ for some u_τ^m which represents some average value for the expected values for u_τ.

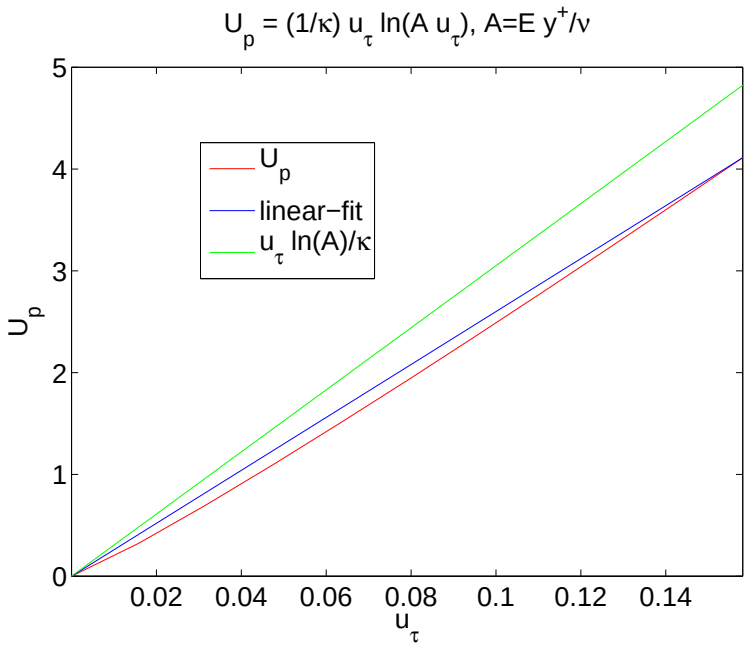


Figure 3: A plot of U_p versus u_τ for the law-of-the wall function, $U_p = \kappa^{-1} u_\tau \ln(A u_\tau)$.

9.3 Baldwin-Lomax Zero Equation Model

The Baldwin-Lomax zero equation model is a two layer model [14]. The inner and outer layers are given by

$$\nu_T = \begin{cases} \nu_{T_i} = \rho l_{\text{mix}}^2 |\omega| & \text{if } y < y_c \\ \nu_{T_o} = \rho\alpha C_{cp} F_{\text{wake}} F_{\text{Kleb}}(y; y_{\max}/C_{\text{Kleb}}) & \text{if } y > y_c \end{cases},$$

where

$$\begin{aligned} y_c &= \min y \ \text{ such that } \ \nu_{T_i} = \nu_{T_o}, \\ l_{\text{mix}} &= \kappa y\Big[1 - e^{-y^+/A_0^+}\Big], \\ F_{\text{wake}} &= \min\left[y_{\max}F_{\max}; C_{wk} y_{\max} U_{\text{dif}}^2/F_{\max}\right] \\ F_{\text{Kleb}}(y;\delta) &= [1 + 5.5(y/\delta)^6]^{-1} \qquad \text{(Klebanoff intermittency function)} \end{aligned}$$

and where $y_{\max}$ and $F_{\max}$ are determined from the maximum of the function

$$F(y) = y\omega\Big[1 - e^{-y^+/A_0^+}\Big]$$

and ω is the magnitude of the vorticity,

$$\begin{aligned} \omega^2 &= (v_x - u_y)^2 + (w_y - v_z)^2 + (u_z - w_x)^2 \\ &= 2\Omega_{ij}\Omega_{ij} \\ \Omega_{ij} &= \frac{1}{2}\Big(\partial u_i/\partial x_j - \partial u_j/\partial x_i\Big) \end{aligned}$$

For boundary layer flows U_{dif} is the maximum value of $\|\mathbf{u}\|$ through the layer?? For more general flows it is

$$U_{\text{dif}} = \Big(\|\mathbf{u}\|\Big)_{\max} - \Big(\|\mathbf{u}\|\Big)\Big|_{y=y_{\max}}$$

Note comment in Wilcox that U_{dif} is NOT the difference between the max and min velocities as specified in the original BL paper and at `http://www.cfd-online.com/Wiki/Baldwin-Lomax_model`

The model parameters are given by

$$\begin{aligned} \kappa = .40, \quad \alpha = 0.0168, \quad A_0^+ = 26 \\ C_{cp} = 1.6, \quad C_{\text{Kleb}} = 0.3 \quad C_{wk} = 1 \text{(or .25 from cfd-online)} \end{aligned}$$

9.4 Spalart-Allmaras turbulence model

Spalart-Allmaras **one equation model**

$$\begin{aligned} \nu_T &= \tilde{\nu} f_{v1} \\ \partial_t\tilde{\nu} + U_j\partial_j\tilde{\nu} &= c_{b1}\tilde{S}\tilde{\nu} - c_{w1}f_w(\tilde{\nu}/d)^2 + \frac{1}{\sigma}\Big[\partial_k[(\nu+\tilde{\nu})\partial_k\tilde{\nu}] + c_{b2}\partial_k\tilde{\nu}\partial_k\tilde{\nu}\Big] \end{aligned}$$

$$c_{b1} = .1355, c_{b2} = .622, c_{v1} = 7.1, \sigma = 2/3$$

$$c_{w1} = \frac{c_{b1}}{\kappa^2} + \frac{(1+c_{b2})}{\sigma}, \quad c_{w2} = 0.3, \quad c_{w3} = 2, \quad \kappa = .41$$

$$f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3}, \quad f_{v2} = 1 - \frac{\chi}{1+\chi f_{v1}}, f_w = g\left[\frac{1+c_{w3}^6}{g^6 + c_{w3}^6}\right]^{1/6}$$

$$\chi = \frac{\tilde{\nu}}{\nu}, \quad g = r + c_{w2}(r^6 - r), \qquad r = \frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2}$$

$$\tilde{S} = S + \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2}, \qquad S = \sqrt{2\Omega_{ij}\Omega_{ij}}$$

$$\Omega_{ij} = (1/2)(\partial_i U_j - \partial_j U_i) \qquad \text{rotation tensor}$$

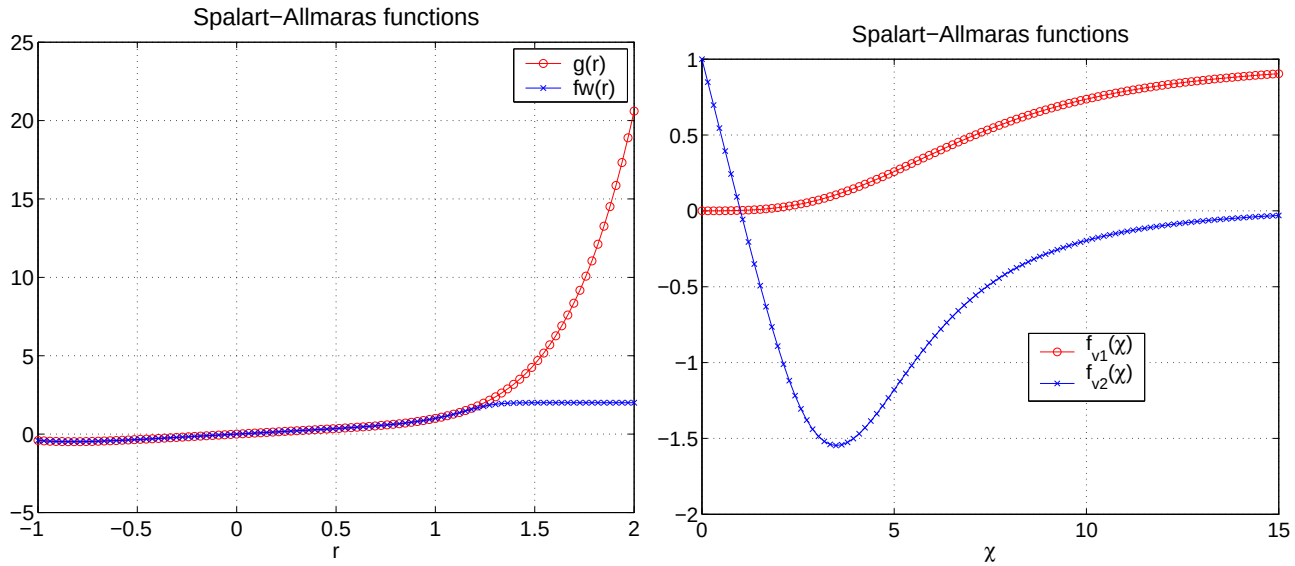


Figure 4: Behaviour of the SpalartConvergence fudge functions

Depends on d, the distance to the nearest surface.

Notes f_{v2} can be positive or negative but is bounded from above by 1 and below by ??

$$\begin{aligned} f_{v2} &= 1 - \frac{\chi}{1+\chi f_{v1}} \\ &= 1 - \frac{1}{\chi^{-1} + 1/[1+(c_{v1}/\chi)^{-3}]} \\ &\to 0 \qquad \text{as } \chi \to \infty \\ &\to 1 \qquad \text{as } \chi \to 0 \\ &\approx 1 - \frac{1}{(1/7)+(1/2)} = -3/4 \qquad \text{when } \chi = c_{v1} \end{aligned}$$

Since f_{v2} can be negative, so can r and g.

On a rectangular grid this is discretized as

$$\begin{aligned} \partial_t \tilde{\nu}_{\mathbf{i}} + U_{\mathbf{i}} D_{0x} U_{\mathbf{i}} + V_{\mathbf{i}} D_{0y} U_{\mathbf{i}} = c_{b1} \tilde{S}_{\mathbf{i}} \tilde{\nu}_{\mathbf{i}} - c_{w1} f_w (\tilde{\nu}/d)^2 \\ + \frac{1}{\sigma} D_{+x}[(\nu + \tilde{\nu}_{i_1 - \frac{1}{2}}) D_{-x} \tilde{\nu}_{\mathbf{i}} + D_{+y}[(\nu + \tilde{\nu}_{i_2 - \frac{1}{2}}) D_{-y} \tilde{\nu}_{\mathbf{i}}] \\ + c_{b2} \Big\{ (D_{0x}\tilde{\nu})^2 + (D_{0y}\tilde{\nu})^2 \Big\} \end{aligned}$$

On curvilinear grids we use the conservative form of the second order term

$$\nabla \cdot (a \nabla \phi) = \frac{1}{J} \Big\{ \frac{\partial}{\partial r_1} \left(A^{11} \frac{\partial \phi}{\partial r_1} \right) + \frac{\partial}{\partial r_2} \left(A^{22} \frac{\partial \phi}{\partial r_2} \right) + \frac{\partial}{\partial r_1} \left(A^{12} \frac{\partial \phi}{\partial r_2} \right) + \frac{\partial}{\partial r_2} \left(A^{21} \frac{\partial \phi}{\partial r_1} \right) \Big\}$$

where

$$\begin{aligned} A^{11} &= aJ \left[\frac{\partial r_1}{\partial x_1}^2 + \frac{\partial r_1}{\partial x_2}^2 \right] \\ A^{22} &= aJ \left[\frac{\partial r_2}{\partial x_1}^2 + \frac{\partial r_2}{\partial x_2}^2 \right] \\ A^{12} &= aJ \left[\frac{\partial r_1}{\partial x_1} \frac{\partial r_2}{\partial x_1} + \frac{\partial r_1}{\partial x_2} \frac{\partial r_2}{\partial x_2} \right] \end{aligned}$$

A **second-order accurate** compact discretization to this expression is

$$\nabla\cdot(a\nabla\phi) \approx \frac{1}{J}\Big\{ D_{+r_1}\left(A^{11}_{i_1-\frac{1}{2}} D_{-r_1}\phi\right) + D_{+r_2}\left(A^{22}_{i_2-\frac{1}{2}} D_{-r_2}\phi\right) + D_{0r_1}\left(A^{12} D_{0r_2}\phi\right) + D_{0r_2}\left(A^{21} D_{0r_1}\phi\right)\Big\}$$

where we can define the cell average values for A^{mn} by

$$A^{11}_{i_1-\frac{1}{2}} \approx \frac{1}{2}(A^{11}_{i_1} + A^{11}_{i_1-1})$$
$$A^{22}_{i_2-\frac{1}{2}} \approx \frac{1}{2}(A^{22}_{i_2} + A^{22}_{i_2-1})$$

9.5 $k-\epsilon$ turbulence model

Here is the $k-\epsilon$ model

$$\begin{aligned}
\nu_T &= C_\mu k^2/\epsilon \\
\partial_t k + U_j\partial_j k &= \tau_{ij}\partial_j U_i - \epsilon + \partial_j[(\nu+\nu_T/\sigma_k)\partial_j k] \\
\partial_t \epsilon + U_j\partial_j \epsilon &= C_{\epsilon 1}\frac{\epsilon}{k}\tau_{ij}\partial_j U_i - C_{\epsilon 2}\epsilon^2/k + \partial_j[(\nu+\nu_T/\sigma_\epsilon)\partial_j \epsilon] \\
C_{\epsilon 1} = 1.44, \quad C_{\epsilon 2} = 1.92, \quad C_\mu &= .09, \quad \sigma_k = 1, \quad \sigma_\epsilon = 1.3
\end{aligned}$$

The production term is $P = \tau_{ij}\partial_j U_i$

$$\begin{aligned}
P &= \tau_{ij}\partial_j U_i \\
&= \nu_T(\partial_j U_i + \partial_i U_j)\partial_j U_i \\
&= \frac{\nu_T}{2}(\partial_j U_i + \partial_i U_j)(\partial_j U_i + \partial_i U_j) \\
&= \frac{\nu_T}{2}\Big((2u_x)^2 + (2v_y)^2 + (2w_z)^2 + 2(u_y+v_x)^2 + 2(u_z+w_x)^2 + 2(v_z+w_y)^2\Big) \\
&= \nu_T\Big(2(u_x^2+v_y^2+w_z^2) + (u_y+v_x)^2 + (u_z+w_x)^2 + (v_z+w_y)^2\Big)
\end{aligned}$$

9.6 Diffusion Operator

When a turbulence model is added to the incompressible Navier-Stokes equation the diffusion operator usually takes the form of

$$\mathcal{D}_i = \sum_j \partial_{x_j}\Big(\nu_T(\partial_{x_i}u_j + \partial_{x_j}u_i)\Big)$$

where we will write ν_T instead of $\nu+\nu_T$ in this section. In particular

$$\begin{aligned}
\mathcal{D}_u &= \partial_x(2\nu_T u_x) + \partial_y(\nu_T u_y) + \partial_z(\nu_T u_z) + \partial_y(\nu_T v_x) + \partial_z(\nu_T w_x) \\
\mathcal{D}_v &= \partial_x(\nu_T v_x) + \partial_y(2\nu_T v_y) + \partial_z(\nu_T v_z) + \partial_x(\nu_T u_y) + \partial_z(\nu_T w_y) \\
\mathcal{D}_w &= \partial_x(\nu_T w_x) + \partial_y(\nu_T w_y) + \partial_z(2\nu_T w_z) + \partial_y(\nu_T v_z) + \partial_x(\nu_T u_z)
\end{aligned}$$

We can write these in a more "symmetric" form as follows. Since $u_x + v_y + w_z = 0$, it follows that

$$\partial_x(\nu_T u_x) = -\partial_x(\nu_T v_y) - \partial_x(\nu_T w_z)$$

and thus

$$\mathcal{D}_u = \partial_x(\nu_T u_x) + \partial_y(\nu_T u_y) + \partial_z(\nu_T u_z) + \partial_y(\nu_T v_x) - \partial_x(\nu_T v_y) + \partial_z(\nu_T w_x) - \partial_x(\nu_T w_z)$$

Therefore the diffusion operator can be written in a form where the principle part is the same for all components,

$$\begin{aligned}
\mathcal{D}_u &= \nabla\cdot(\nu_T\nabla u) + \partial_y(\nu_T v_x) - \partial_x(\nu_T v_y) + \partial_z(\nu_T w_x) - \partial_x(\nu_T w_z)\\
\mathcal{D}_v &= \nabla\cdot(\nu_T\nabla v) + \partial_z(\nu_T w_y) - \partial_y(\nu_T w_z) + \partial_x(\nu_T u_y) - \partial_y(\nu_T u_x)\\
\mathcal{D}_w &= \nabla\cdot(\nu_T\nabla w) + \partial_x(\nu_T u_z) - \partial_z(\nu_T u_x) + \partial_y(\nu_T v_z) - \partial_z(\nu_T v_y)
\end{aligned}$$

Note that the highest order derivatives cancel in the last four terms in these expressions,

$$\begin{aligned}
\mathcal{D}_u &= \nabla\cdot(\nu_T\nabla u) + \partial_y(\nu_T) v_x - \partial_x(\nu_T) v_y + \partial_z(\nu_T) w_x - \partial_x(\nu_T) w_z\\
\mathcal{D}_v &= \nabla\cdot(\nu_T\nabla v) + \partial_z(\nu_T) w_y - \partial_y(\nu_T) w_z + \partial_x(\nu_T) u_y - \partial_y(\nu_T) u_x\\
\mathcal{D}_w &= \nabla\cdot(\nu_T\nabla w) + \partial_x(\nu_T) u_z - \partial_z(\nu_T) u_x + \partial_y(\nu_T) v_z - \partial_z(\nu_T) v_y
\end{aligned}$$

9.7 Revised pressure equation

When a turbulence model is added to the incompressible Navier-Stokes equation the diffusion operator usually takes the form of

$$\mathcal{D}_i = \sum_j \partial_{x_j}\Big(\nu_T(\partial_{x_i}u_j + \partial_{x_j}u_i)\Big)$$

The pressure equation is derived from taking the divergence of the momentum equations,

$$\partial_t\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \mathcal{D}$$

and using $\nabla\cdot\mathbf{u} = 0$ to give

$$\Delta p = -\nabla\mathbf{u} : \nabla\mathbf{u} + \nabla\cdot\mathcal{D}$$

For a constant viscosity, the last term on the right hand side is zero. When the viscosity is not constant we need to include the divergence of the diffusion operator in the equation for the pressure. This takes the form

$$\begin{aligned}
\nabla\cdot\mathcal{D} &= \sum_i\sum_j \partial_{x_i}\partial_{x_j}\Big[\nu_T\Big(\partial_{x_j}u_i + \partial_{x_i}u_j\Big)\Big]\\
&= \sum_j \partial_{x_j}\Big[\sum_i \partial_{x_i}\nu_T\ \partial_{x_j}u_i\Big] + \sum_i \partial_{x_i}\Big[\sum_j \partial_{x_j}\nu_T\ \partial_{x_i}u_j\Big]\\
&= 2\ \sum_i \partial_{x_i}\Big[\sum_j\ \partial_{x_j}\nu_T\ \partial_{x_i}u_j\Big]
\end{aligned}$$

where we have again used $\nabla\cdot\mathbf{u} = 0$.

In two dimensions this takes the form

$$\begin{aligned}
\nabla\cdot\mathcal{D}^{(2\mathrm{d})} &= 2\ \Big[\partial_x\Big(\partial_x\nu_T\ \partial_x u + \partial_y\nu_T\ \partial_x v\Big) + \partial_y\Big(\partial_x\nu_T\ \partial_y u + \partial_y\nu_T\ \partial_y v\Big)\Big]\\
&= 2\ \Big[\partial_x\nu_T\ \Delta u + \partial_x^2\nu_T\ \partial_x u + \partial_x\partial_y\nu_T\ \partial_y u\\
&\qquad + \partial_y\nu_T\ \Delta v + \partial_x\partial_y\nu_T\ \partial_x v + \partial_y^2\nu_T\ \partial_y v\Big]
\end{aligned}$$

In three dimensions,

$$\begin{aligned}\nabla\cdot\mathcal{D}^{(3\mathrm{d})} &= 2\,\Big[\partial_x\Big(\partial_x\nu_T\;\partial_x u+\partial_y\nu_T\;\partial_x v+\partial_z\nu_T\;\partial_x w\Big)\\ &\quad+\partial_y\Big(\partial_x\nu_T\;\partial_y u+\partial_y\nu_T\;\partial_y v+\partial_z\nu_T\;\partial_y w\Big)\\ &\quad+\partial_z\Big(\partial_x\nu_T\;\partial_z u+\partial_y\nu_T\;\partial_z v+\partial_z\nu_T\;\partial_z w\Big)\Big]\\ &= 2\,\Big[\partial_x\nu_T\;\Delta u+\partial_x^2\nu_T\;\partial_x u+\partial_x\partial_y\nu_T\;\partial_y u+\partial_x\partial_z\nu_T\;\partial_z u\\ &\quad+\partial_y\nu_T\;\Delta v+\partial_x\partial_y\nu_T\;\partial_x v+\partial_y^2\nu_T\;\partial_y v+\partial_y\partial_z\nu_T\;\partial_z v\\ &\quad+\partial_z\nu_T\;\Delta w+\partial_x\partial_z\nu_T\;\partial_x w+\partial_y\partial_z\nu_T\;\partial_y w+\partial_z^2\nu_T\;\partial_z w\Big]\end{aligned}$$

The addition of artificial dissipation also changes the pressure equation. The second-order artificial dissipation is

$$\mathbf{d}_{2,i}=(\texttt{ad21}+\texttt{ad22}|\nabla_h\mathbf{V}_i|_1)\sum_{m=1}^{n_d}\Delta_{m+}\Delta_{m-}\mathbf{V}_i \tag{46}$$

while the the fourth-order one is

$$\mathbf{d}_{4,i}=-(\texttt{ad41}+\texttt{ad42}|\nabla_h\mathbf{V}_i|_1)\sum_{m=1}^{n_d}\Delta_{m+}^2\Delta_{m-}^2\mathbf{V}_i \tag{47}$$

The artificial dissipation is of the form

$$\begin{aligned}\mathbf{d}&=\Big[\alpha_0+\alpha_1\mathcal{G}(\nabla\mathbf{u})\Big]\sum_{m=1}^{n_d}\Delta_{m+}^p\Delta_{m-}^p\mathbf{u}\\ \mathcal{G}(\nabla\mathbf{u})&=|u_x|+|u_y|+|v_x|+|v_y|+\ldots\end{aligned}$$

Taking the divergence of this expression results in

$$\nabla\cdot\mathbf{d}=\alpha_1\Big[\mathcal{G}_x\sum_m\Delta_{m+}^p\Delta_{m-}^pU+\mathcal{G}_y\sum_m\Delta_{m+}^p\Delta_{m-}^pV\Big]$$

where

$$\mathcal{G}_x=\mathrm{sgn}(u_x)\;u_{xx}+\mathrm{sgn}(u_y)\;u_{xy}+\mathrm{sgn}(v_x)\;v_{xx}+\mathrm{sgn}(v_y)\;v_{xy}+\ldots$$

and $\mathrm{sgn}(x)$ is $+1$, -1 or 0 for $x>0$, $x<0$ or $x=0$.

9.7.1 Revised pressure boundary condition

The *pressure boundary condition* also is changed to include the new diffusion operator:

$$\partial_n p=\mathbf{n}\cdot\Big\{-\partial_t\mathbf{u}-(\mathbf{u}\cdot\nabla)\mathbf{u}+\mathcal{D}\Big\}$$

We would like to write the diffusion operator in a way similiar to *curl-curl* form,

$$\Delta\mathbf{u}=-\nabla\times\nabla\times\mathbf{u}+\nabla(\nabla\cdot\mathbf{u}),$$

used in the INS equations. Expanding the expression for $\mathcal{D}_u$ gives

$$\begin{aligned}\mathcal{D}_u&=\partial_x(2\nu_Tu_x)+\partial_y(\nu_Tu_y)+\partial_z(\nu_Tu_z)+\partial_y(\nu_Tv_x)+\partial_z(\nu_Tw_x)\\ &=2\nu_Tu_{xx}+\nu_Tu_{yy}+\nu_Tu_{zz}+2\partial_x\nu_Tu_x+\partial_y\nu_T(u_y+v_x)+\partial_z\nu_T(u_z+w_x)+\nu_T(v_{xy}+w_{xz})\\ &=\nu_T\Delta u+2\partial_x\nu_Tu_x+\partial_y\nu_T(u_y+v_x)+\partial_z\nu_T(u_z+w_x)\end{aligned}$$

where we have used $v_{xy} + w_{xz} = -u_{xx}$ Thus we can write

$$\mathcal{D}_u = \nu_T \Delta u - 2\partial_x \nu_T (v_y + w_z) + \partial_y \nu_T (u_y + v_x) + \partial_z \nu_T (u_z + w_x)$$

This leads in two dimensions to the *curl-curl* form

$$\mathcal{D}_u^{(2D)} = \nu_T(-v_{xy} + u_{yy}) - 2\partial_x \nu_T v_y + \partial_y \nu_T (u_y + v_x) \tag{48}$$

$$\mathcal{D}_v^{(2D)} = \nu_T(\ \ v_{xx} - u_{xy}) - 2\partial_y \nu_T u_x + \partial_x \nu_T (v_x + u_y) \tag{49}$$

while in three dimensions,

$$\mathcal{D}_u = \nu_T(-v_{xy} - w_{xz} + u_{yy} + u_{zz}) - 2\partial_x \nu_T (v_y + w_z) + \partial_y \nu_T (u_y + v_x) + \partial_z \nu_T (u_z + w_x) \tag{50}$$

$$\mathcal{D}_v = \nu_T(\ \ v_{xx} - u_{xy} - w_{yz} + v_{zz}) - 2\partial_y \nu_T (w_z + u_x) + \partial_z \nu_T (v_z + w_y) + \partial_x \nu_T (v_x + u_y) \tag{51}$$

$$\mathcal{D}_w = \nu_T(\ \ w_{xx} + w_{yy} - u_{xz} - v_{yz}) - 2\partial_z \nu_T (u_x + v_y) + \partial_x \nu_T (w_x + u_z) + \partial_y \nu_T (w_y + v_z) \tag{52}$$

These *curl-curl* forms (48-52) remove the normal derivatives of the normal components of the velocity from $\mathbf{n} \cdot (\mathcal{D}_u, \mathcal{D}_v, \mathcal{D}_w)$. For example, the expression for $\mathcal{D}_u$ contains no x-derivatives of u while $\mathcal{D}_v$ contains no y-derivatives of v.

The alternative conservative form is

$$\begin{aligned}
\mathcal{D}_u &= \partial_x(-2\nu_T(v_y + w_z)) + \partial_y(\nu_T u_y) + \partial_z(\nu_T u_z) + \partial_y(\nu_T v_x) + \partial_z(\nu_T w_x) \\
\mathcal{D}_v &= \partial_x(\nu_T v_x) + \partial_y(-2\nu_T(u_x + w_z)) + \partial_z(\nu_T v_z) + \partial_x(\nu_T u_y) + \partial_z(\nu_T w_y) \\
\mathcal{D}_w &= \partial_x(\nu_T w_x) + \partial_y(\nu_T w_y) + \partial_z(-2\nu_T(u_x + v_y)) + \partial_y(\nu_T v_z) + \partial_x(\nu_T u_z)
\end{aligned}$$

10 Steady state line solver

We first consider the case of a rectangular grid in two space dimensions.

The implicit line solver uses local time stepping where the local time step $\Delta_{\mathbf{i}}$ is defined from

$$\Delta t_{\mathbf{i}} = ...$$

We solve implicit scalar-tri-diagonal systems in each spatial direction. Along the x-direction we solve a tridiagonal system for U, followed by a tridiagonal system for V for the equations

$$\begin{aligned}
\frac{U_{\mathbf{i}}^{n+1} - U_{\mathbf{i}}^{n}}{\Delta t_{\mathbf{i}}} &= -\Big\{ U^n D_{0x} U^{n+1} + V^n D_{0y} U^n + D_{0x} P^n \Big\} \\
&\quad + \nu \Big\{ D_{+x} D_{-x} U^{n+1} + (U_{j+1}^n - 2U^{n+1} + U_{j-1}^n)/h_y^2 \Big\} \\
&\quad + \nu_A(\mathbf{U}^n) \Big\{ \Delta_{+x} \Delta_{-x} U^{n+1} + (U_{j+1}^n - 2U^{n+1} + U_{j-1}^n) \Big\} \\
\frac{V_{\mathbf{i}}^{n+1} - V_{\mathbf{i}}^{n}}{\Delta t_{\mathbf{i}}} &= -\Big\{ U^n D_{0x} V^{n+1} + V^n D_{0y} V^n + D_{0y} P^n \Big\} \\
&\quad + \nu \Big\{ D_{+x} D_{-x} V^{n+1} + (V_{j+1}^n - 2V^{n+1} + V_{j-1}^n)/h_y^2 \Big\} \\
&\quad + \nu^{(2)}(\mathbf{U}^n) \Big\{ \Delta_{+x} \Delta_{-x} V^{n+1} + (V_{j+1}^n - 2V^{n+1} + V_{j-1}^n) \Big\}
\end{aligned}$$

Here $\nu^{(2)}(\mathbf{U}^n)$ is the coefficient of the artificial dissipation. There is also a self-adjoint version of the artificial dissipation,

$$\begin{aligned}
\beta_{SA} &= \Delta_{+x} \Big[\nu^{(2)}_{i_1 - \frac{1}{2}} \Delta_{-x} \Big] U + \Delta_{+y} \Big[\nu^{(2)}_{i_2 - \frac{1}{2}} \Delta_{-y} \Big] U \\
&= \nu^{(2)}_{i_1 + \frac{1}{2}} (U_{i_1+1} - U) - \nu^{(2)}_{i_1 - \frac{1}{2}} (U - U_{i_1 - 1}) \\
&+ \nu^{(2)}_{i_2 + \frac{1}{2}} (U_{i_2+1} - U) - \nu^{(2)}_{i_2 - \frac{1}{2}} (U - U_{i_2 - 1}) \\
&= \nu^{(2)}_{i_1 + \frac{1}{2}} U_{i_1+1} + \nu^{(2)}_{i_1 - \frac{1}{2}} U_{i_1 - 1} + \nu^{(2)}_{i_2 + \frac{1}{2}} U_{i_2+1} + \nu^{(2)}_{i_2 - \frac{1}{2}} U_{i_2 - 1} \\
&- \Big(\nu^{(2)}_{i_1 + \frac{1}{2}} + \nu^{(2)}_{i_1 - \frac{1}{2}} + \nu^{(2)}_{i_2 + \frac{1}{2}} + \nu^{(2)}_{i_2 - \frac{1}{2}} \Big) U
\end{aligned}$$

After solving in the x-direction we then solve along lines in the y-direction.

On curvilinear grids the expressions are a bit more complicated. Before discretization the equations transformed to the unit-square are

$$\begin{aligned}
u_t &= -\Big\{ (u r_x + v r_y) u_r + (u s_x + v s_y) u_s + r_x p_r + s_x p_s \Big\} \\
&\quad + \nu \frac{1}{J} \Big\{ \partial_r (J(r_x u_x + r_y u_y)) + \partial_s (J(s_x u_x + s_y u_y)) \Big\} \\
u_x &= r_x u_r + s_x u_s \\
u_y &= r_y u_r + s_y u_s \\
J &= |\partial \mathbf{x} / \partial \mathbf{r}| = x_r y_s - x_s y_r
\end{aligned}$$

We solve scalar-tridiagonal-systems in the r and s directions.

10.1 Fourth-order artificial dissipation

A fourth-order artificial dissipation is

$$\nu^{(4)}(\mathbf{U}^n)\Big\{(\Delta_{+x}\Delta_{-x})^2U + (\Delta_{+y}\Delta_{-y})^2U\Big\}$$

or in a self-adjoint form

$$\begin{aligned}
\beta^{(4)}_{SA} &= \Delta_{+x}\Delta_{-x}\Big[\nu^{(4)}_{\mathbf{i}}\Delta_{+x}\Delta_{-x}\Big]U + \Delta_{+y}\Delta_{-y}\Big[\nu^{(4)}_{\mathbf{i}}\Delta_{+y}\Delta_{-y}\Big]U \\
&= \nu^{(4)}_{i_1+1}\Delta_{+x}\Delta_{-x}U_{i_1+1} - 2\nu^{(4)}_{\mathbf{i}}\Delta_{+x}\Delta_{-x}U + \nu^{(4)}_{i_1-1}\Delta_{+x}\Delta_{-x}U_{i_1-1} \\
&+ \nu^{(4)}_{i_2+1}\Delta_{+y}\Delta_{-y}U_{i_2+1} - 2\nu^{(4)}_{\mathbf{i}}\Delta_{+y}\Delta_{-y}U + \nu^{(4)}_{i_2-1}\Delta_{+y}\Delta_{-y}U_{i_2-1} \\
&= \nu^{(4)}_{i_1+1}U_{i_1+2} - 2\big[\nu^{(4)}_{i_1+1} + \nu^{(4)}_{\mathbf{i}}\big]U_{i_1+1} + \nu^{(4)}_{i_1-1}U_{i_1-2} - 2\big[\nu^{(4)}_{i_1-1} + \nu^{(4)}_{\mathbf{i}}\big]U_{i_1-1} \\
&+ \nu^{(4)}_{i_2+1}U_{i_2+2} - 2\big[\nu^{(4)}_{i_2+1} + \nu^{(4)}_{\mathbf{i}}\big]U_{i_2+1} + \nu^{(4)}_{i_2-1}U_{i_2-2} - 2\big[\nu^{(4)}_{i_2-1} + \nu^{(4)}_{\mathbf{i}}\big]U_{i_2-1} \\
&+ \big[\nu^{(4)}_{i_1+1} + \nu^{(4)}_{i_2+1} + 8\nu^{(4)}_{\mathbf{i}} + \nu^{(4)}_{i_1-1} + \nu^{(4)}_{i_2-1}\big]U_{\mathbf{i}}
\end{aligned}$$

11 Convergence results

This section details the results of various convergence tests. Convergence results are run using the **twilight-zone** option, also known less formally as the **method of analytic solutions**. In this case the equations are forced so the the solution will be a known analytic function.

The tables show the maximum errors in the solution components. The rate shown is estimated convergence rate, σ, assuming error $\propto h^\sigma$. The rate is estimated by a least squares fit to the data.

The 2D trigonometric solution used as a twilight zone function is

$$\begin{aligned} u &= \frac{1}{2}\cos(\pi\omega_0 x)\cos(\pi\omega_1 y)\cos(\omega_3\pi t) + \frac{1}{2} \\ v &= \frac{1}{2}\sin(\pi\omega_0 x)\sin(\pi\omega_1 y)\cos(\omega_3\pi t) + \frac{1}{2} \\ p &= \cos(\pi\omega_0 x)\cos(\pi\omega_1 y)\cos(\omega_3\pi t) + \frac{1}{2} \end{aligned}$$

The 3D trigonometric solution is

$$\begin{aligned} u &= \cos(\pi\omega_0 x)\cos(\pi\omega_1 y)\cos(\pi\omega_2 z)\cos(\omega_3\pi t) \\ v &= \frac{1}{2}\sin(\pi\omega_0 x)\sin(\pi\omega_1 y)\cos(\pi\omega_2 z)\cos(\omega_3\pi t) \\ w &= \frac{1}{2}\sin(\pi\omega_0 x)\sin(\pi\omega_1 y)\sin(\pi\omega_2 z)\cos(\omega_3\pi t) \\ p &= \frac{1}{2}\sin(\pi\omega_0 x)\cos(\pi\omega_1 y)\cos(\pi\omega_2 z)\sin(\omega_3\pi t) \end{aligned}$$

When $\omega_0 = \omega_1 = \omega_2$ it follows that $\nabla \cdot \mathbf{u} = 0$. There are also algebraic polynomial solutions of different orders.

Tables (2-6) show results from running Cgins on various grids.

grid	N	p	u	v	$\mathbf{u}$	$\nabla \cdot \mathbf{u}$
square20	20	2.9×10^{-1}	3.4×10^{-2}	3.4×10^{-2}	3.4×10^{-2}	5.0×10^{-1}
square30	30	9.6×10^{-2}	1.3×10^{-2}	1.2×10^{-2}	1.3×10^{-2}	1.8×10^{-1}
square40	40	4.5×10^{-2}	7.2×10^{-3}	6.7×10^{-3}	7.2×10^{-3}	8.1×10^{-2}
rate		2.69	2.23	2.34	2.24	2.61

Table 1: incompressible Navier Stokes, order=2, $\nu = 0.1$, $t = 1$, square, trig TZ, $\omega = 5.1$, $\alpha = 1$

grid	N	p	u	v	$\mathbf{u}$	$\nabla \cdot \mathbf{u}$
square16.order4	16	8.3×10^{-2}	1.2×10^{-2}	1.2×10^{-2}	1.2×10^{-2}	1.4×10^{-1}
square32.order4	32	6.5×10^{-3}	4.8×10^{-4}	3.9×10^{-4}	4.8×10^{-4}	7.6×10^{-3}
square64.order4	64	3.4×10^{-4}	2.6×10^{-5}	2.3×10^{-5}	2.6×10^{-5}	2.6×10^{-4}
rate		3.97	4.41	4.50	4.41	4.54

Table 2: incompressible Navier Stokes, order=4, $\nu = 0.1$, $t = 1$, square, trig TZ, $\omega = 5.1$, $\alpha = 1$

grid	N	p	u	v	$\nabla \cdot \mathbf{u}$
cic1	13	2.6×10^{-1}	1.5×10^{-1}	1.6×10^{-1}	5.2×10^{-1}
cic2	25	5.6×10^{-2}	2.4×10^{-2}	2.6×10^{-2}	1.3×10^{-1}
cic3	49	1.5×10^{-2}	5.1×10^{-3}	4.3×10^{-3}	2.3×10^{-2}
cic4	73	3.9×10^{-3}	1.3×10^{-3}	7.4×10^{-4}	4.9×10^{-3}
rate		2.4	2.7	3.0	2.7

Table 3: incompressible Navier Stokes, order=2, $\nu = 0.1$, $t = 1$, cic, trig TZ, $\omega = 2$

grid	N	p	u	v	$\mathbf{u}$	$\nabla \cdot \mathbf{u}$
cicb.order4	61	1.2×10^{-4}	3.8×10^{-5}	3.7×10^{-5}	3.8×10^{-5}	1.9×10^{-4}
cic.order4	121	9.6×10^{-6}	1.7×10^{-6}	2.1×10^{-6}	2.1×10^{-6}	9.6×10^{-6}
cic2.order4	241	6.2×10^{-7}	1.0×10^{-7}	1.2×10^{-7}	1.2×10^{-7}	1.0×10^{-6}
rate		3.86	4.30	4.18	4.19	3.78

Table 4: incompressible Navier Stokes, order=4, $\nu = 0.1$, $t = 1$, cic, trig TZ, $\omega = 1$, $\alpha = 1$

grid	N	p	u	v	w	$\nabla \cdot \mathbf{u}$
box10	10	2.0×10^{-2}	2.0×10^{-3}	2.1×10^{-3}	2.1×10^{-3}	1.7×10^{-2}
box20	20	5.0×10^{-3}	3.4×10^{-4}	3.1×10^{-4}	3.1×10^{-4}	2.7×10^{-3}
box30	30	2.4×10^{-3}	1.3×10^{-4}	1.0×10^{-4}	1.0×10^{-4}	8.1×10^{-4}
rate		1.9	2.5	2.8	2.8	2.8

Table 5: incompressible Navier Stokes, order=2, $\nu = 0.1$, $t = 1$, box, trig TZ, $\omega = 2$

grid	N	p	u	v	w	$\nabla \cdot \mathbf{u}$
box8.order4	8	1.4×10^{-3}	2.9×10^{-4}	2.6×10^{-4}	2.6×10^{-4}	1.2×10^{-3}
box16.order4	16	3.8×10^{-5}	8.4×10^{-6}	7.8×10^{-6}	7.8×10^{-6}	6.6×10^{-5}
box32.order4	32	3.6×10^{-6}	1.7×10^{-7}	1.8×10^{-7}	1.8×10^{-7}	1.6×10^{-6}
rate		4.3	5.4	5.2	5.2	4.8

Table 6: incompressible Navier Stokes, order=4, $\nu = 0.1$, $t = 1$, box, trig TZ, $\omega = 2$

grid	N	p	u	v	w	$\nabla \cdot \mathbf{u}$
sib1	17	2.8×10^{-1}	2.1×10^{-1}	1.4×10^{-1}	1.2×10^{-1}	6.2×10^{-1}
sib2	33	6.2×10^{-2}	5.0×10^{-2}	3.0×10^{-2}	1.9×10^{-2}	1.9×10^{-1}
sib2a	49	3.0×10^{-2}	1.7×10^{-2}	8.8×10^{-3}	1.2×10^{-2}	7.8×10^{-2}
rate		2.1	2.4	2.6	2.2	1.9

Table 7: incompressible Navier Stokes, order=2, $\nu = 0.1$, $t = 1$, sib, trig TZ, $\omega = 2$

grid	N	p	u	v	w	$\mathbf{u}$	$\nabla \cdot \mathbf{u}$
sib1.order4	30	1.0×10^{-2}	1.6×10^{-2}	8.8×10^{-3}	7.5×10^{-3}	1.6×10^{-2}	8.3×10^{-2}
sib1a.order4	60	8.2×10^{-4}	9.9×10^{-4}	4.5×10^{-4}	6.6×10^{-4}	9.9×10^{-4}	6.2×10^{-3}
sib2.order4	80	1.2×10^{-4}	8.7×10^{-5}	5.0×10^{-5}	7.8×10^{-5}	8.7×10^{-5}	7.8×10^{-4}
rate		4.38	5.08	5.10	4.44	5.08	4.57

Table 8: incompressible Navier Stokes, order=4, $\nu = 0.05$, $t = 1$, sib, trig TZ, $\omega = 1$, $\alpha = 1$

grid	N	p	u	v	w	$\mathbf{u}$	$\nabla \cdot \mathbf{u}$
curvedPipe1	1	1.4×10^{-3}	7.0×10^{-4}	1.2×10^{-3}	4.9×10^{-4}	1.2×10^{-3}	4.4×10^{-2}
curvedPipe2	2	4.6×10^{-4}	2.0×10^{-4}	2.3×10^{-4}	1.4×10^{-4}	2.3×10^{-4}	1.5×10^{-2}
curvedPipe3	3	2.6×10^{-4}	9.4×10^{-5}	8.8×10^{-5}	1.1×10^{-4}	1.1×10^{-4}	8.9×10^{-3}
rate		1.51	1.82	2.37	1.39	2.16	1.46

Table 9: INS, curvedPipe, order=2, $\nu = 0.025$, $t = 0.1$, cdv=1, poly TZ

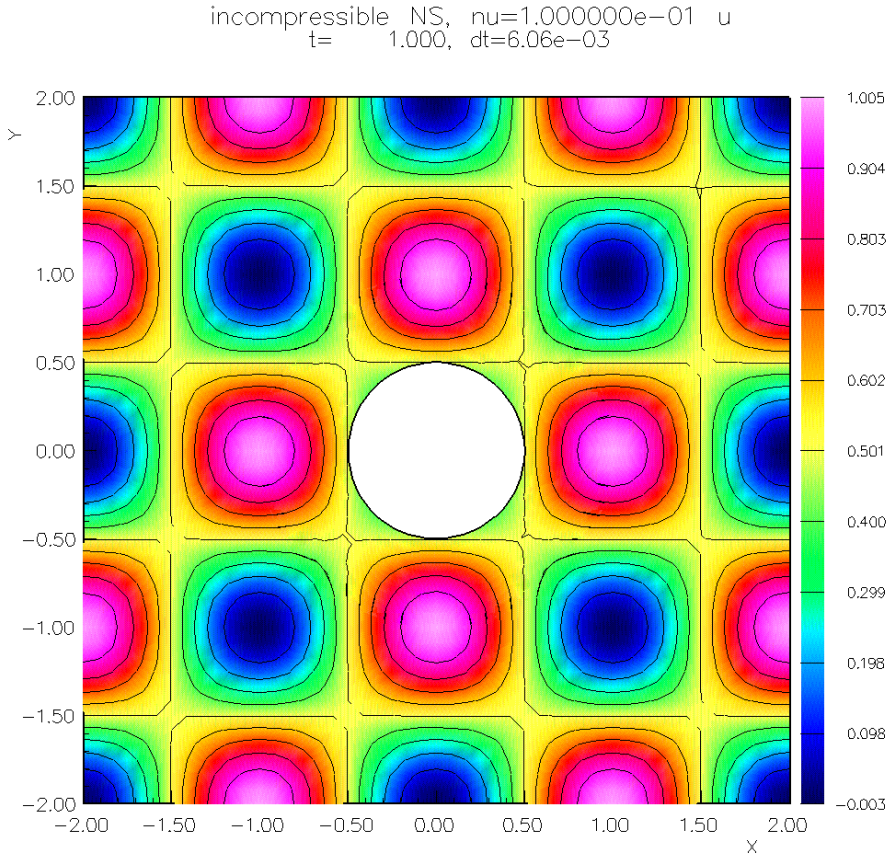


Figure 5: Incompressible N-S, twilight zone solution for convergence test

12 Some sample simulations

Here is a collection of interesting examples computed with the Cgins incompressible solver.

The examples include

- Falling cylinders in an incompressible flow, Section 12.1.
- Flow past two cylinders, Section 12.2.
- A pitching and plunging airofil, Section 12.3.
- A fluttering plat, Section 12.4.
- Simulation of flow past a blood clot filter, Section 12.5.
- Incompressible flow past a truck, Section 12.6.
- Incompressible flow past a city scape, Section 12.7.

12.1 Falling cylinders in an incompressible flow

Figure (6) shows multiple rigid bodies failing under the influence of gravity in an incompressible flow.

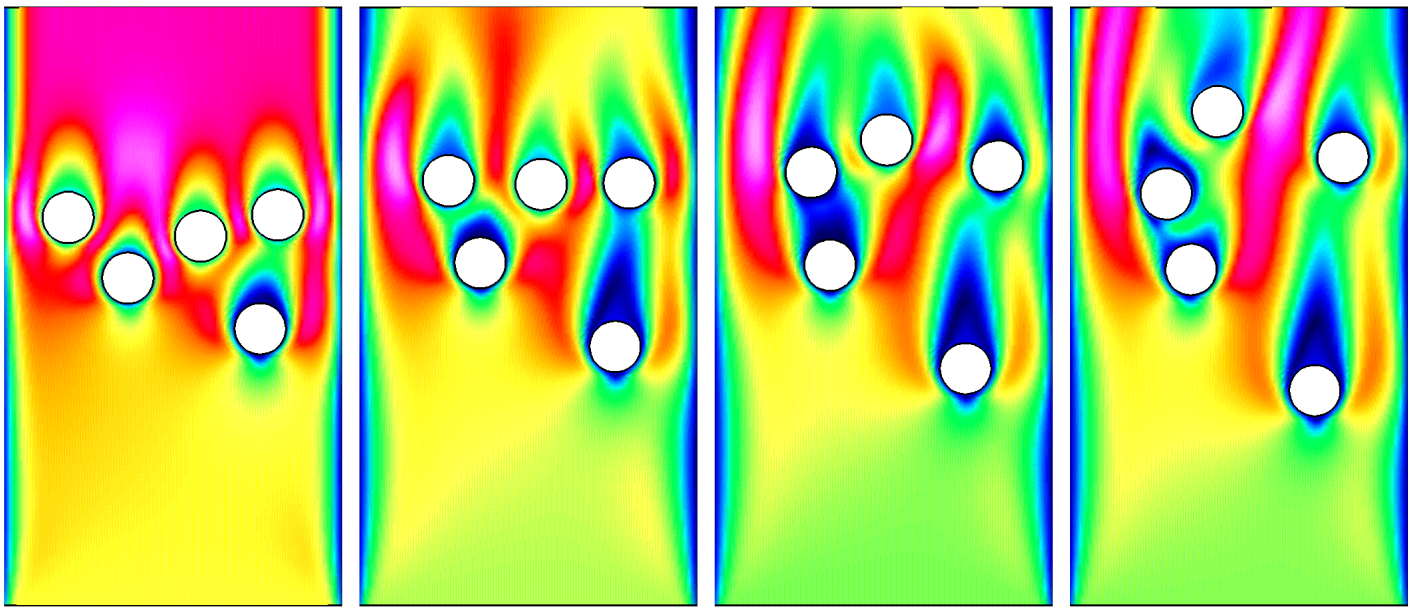


Figure 6: Multiple cylinders falling in an incompressible flow, contour plots of the flow speed. Solution at times $t = 1$, 3, 5 and 7.

There is an upward flow that nearly matches the speed of the falling drops. This solution was computed using the multigrid solver for the pressure equation.

12.2 Flow past two cylinders

Figure 7 shows the simulation of a high Reynolds number flow past two cylinders.

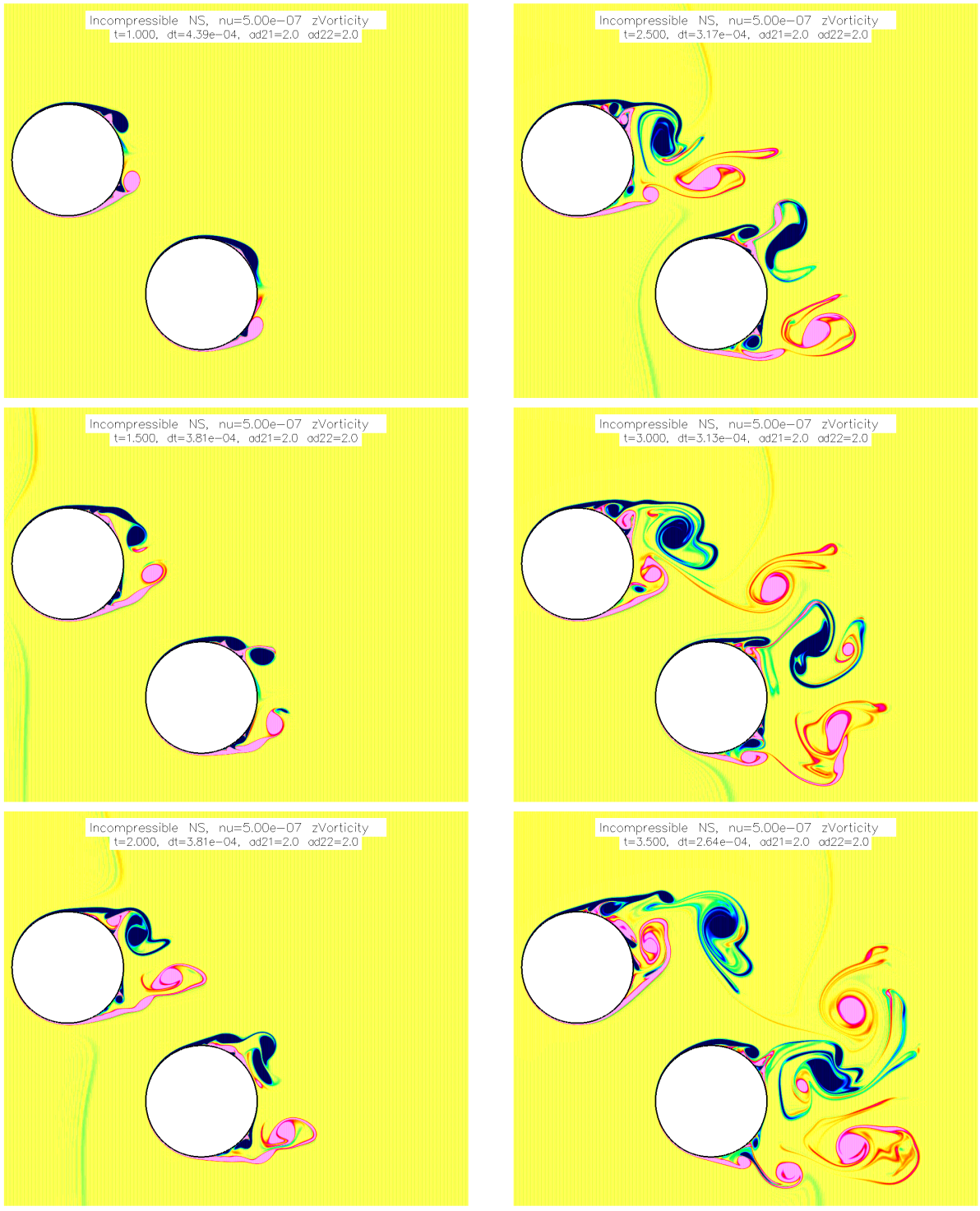


Figure 7: Flow past two cylinders in a channel computed with Cgins. Contour plots of the vorticity.

The simulation used the command file `cg/ins/cmd/tcilc.cmd` and the grid was made with the ogen script `Overture/sampleGrids/tcilc.cmd`. The radius of the cylinders was 1/2. The grid had about 22 million grid points.

12.3 Pitching and plunging airofil

This example shows the simulation of a pitching and plunging airfoil,

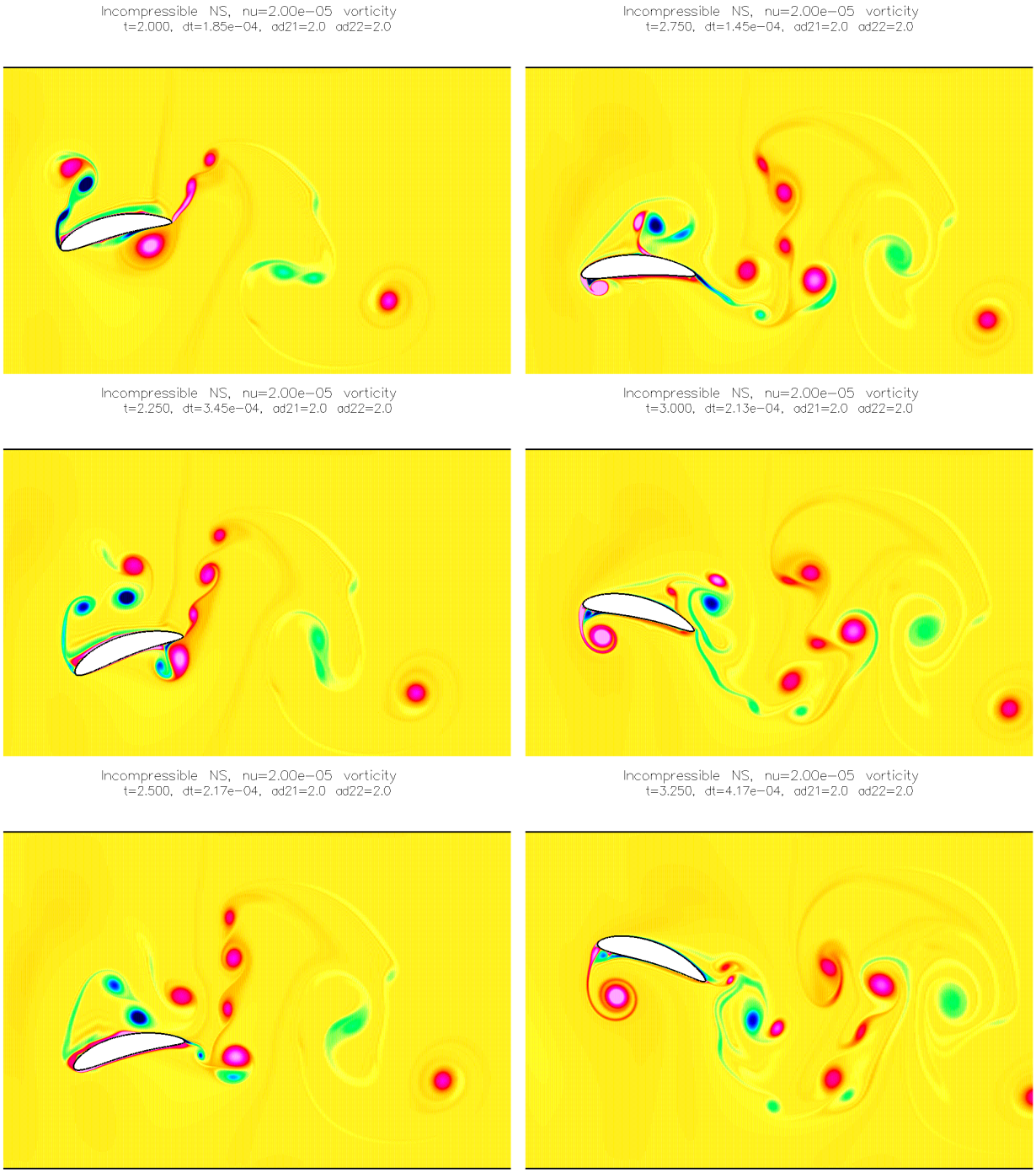


Figure 8: A pitching and plunging airfoil, computed in parallel with Cgins. Contour plots of the vorticity.

The simulation used the command file `cg/ins/cmd/wing2d.cmd` and the grid was made with the ogen script `Overture/sampleGrids/joukowsky2d.cmd`. The computation was performed in parallel and demonstrates the parallel moving grid capabilities (starting with v24). The grid had $1.7M$ points.

12.4 Fluttering plate

This example shows the simulation of a light plate falling under the influence of gravity. There is an upward flow with speed approximately equal to the rate at which the plate falls. The plate starts to sway from side to side and eventually also tumbles.

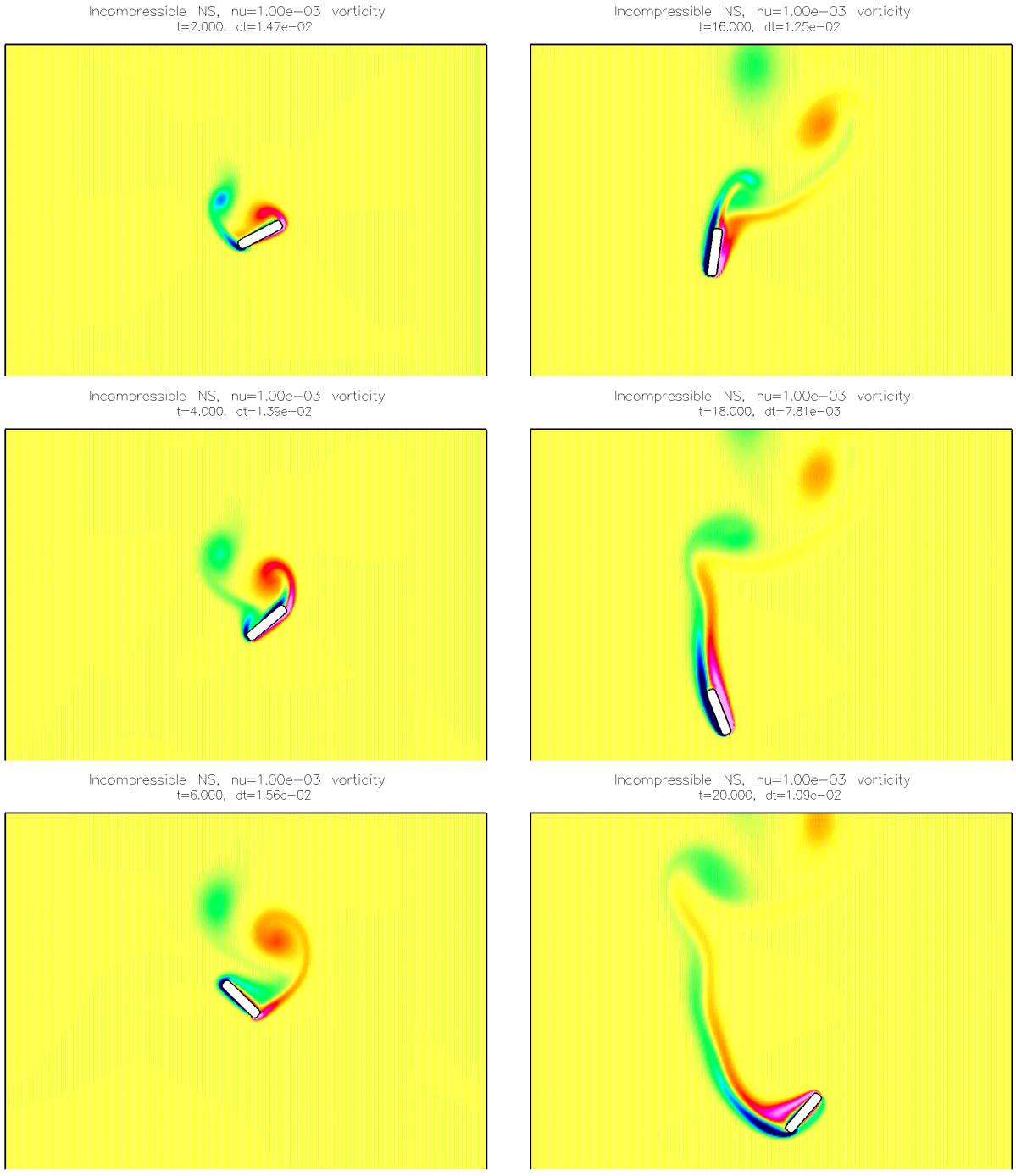


Figure 9: A fluttering plate computed with Cgins. A light plate falling due to the force of gravity in an incompressible flow. Contour plots of the vorticity.

This example demonstrates the new time stepping scheme that has been developed to treat the motion of "light" rigid bodies (v24).

12.5 Simulation of flow past a blood clot filter

Figure 10 shows results from computations of the incompressible flow in a three-dimensional cylindrical pipe ("vein") with an embedded wire frame "filter" and embedded blood clots.

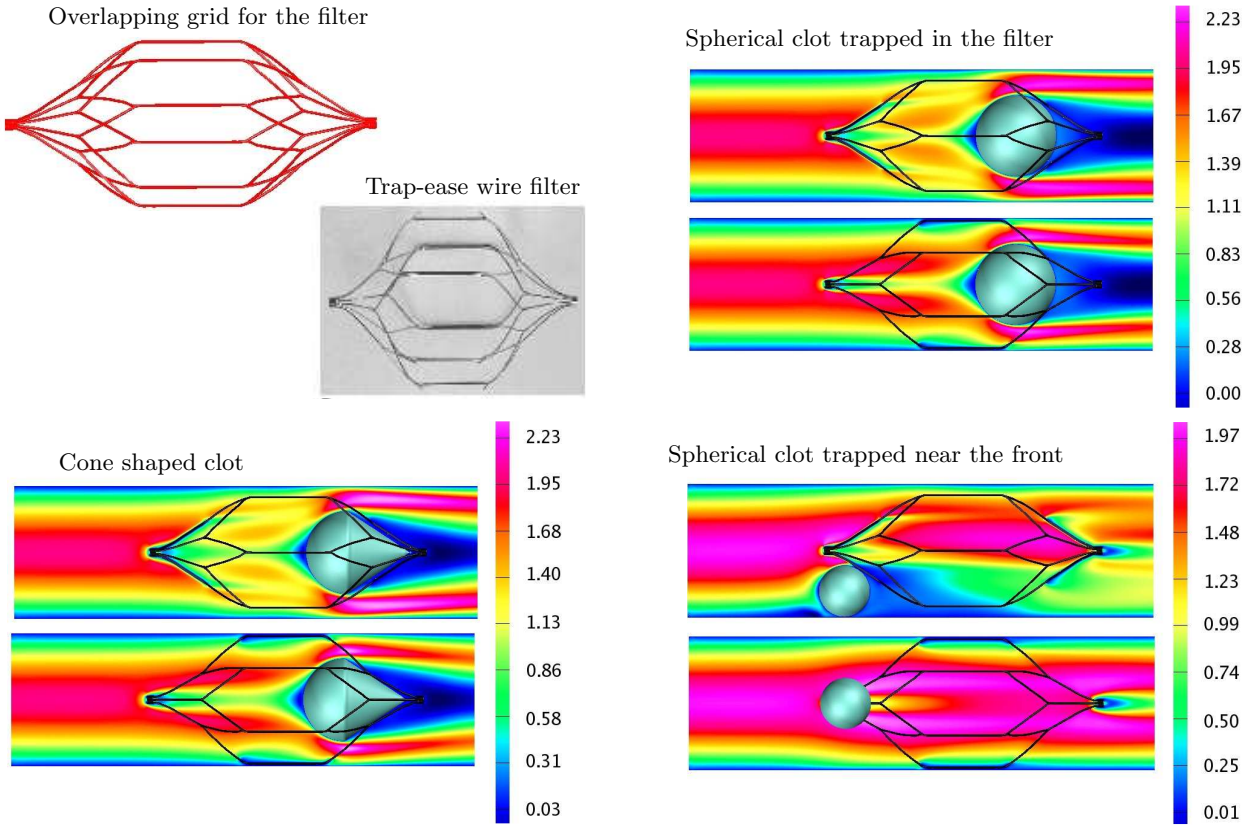


Figure 10: Flow past a blood-clot filter using cgins.

The computations were performed with the pseudo steady-state solution algorithm.

Further details on these simulations can be found in M.A. Singer, WDH, S.L. Wang, *Computational Modeling of Blood Flow in the Trapease Inferior Vena Cava Filter*, Journal of Vascular and Interventional Radiology, **20**, 2009.

12.6 Incompressible flow past a truck

Figure (11) shows a computation of the incompressible Navier-Stokes equations for flow past the cab of a truck. The steady state line solver was used for this computation.

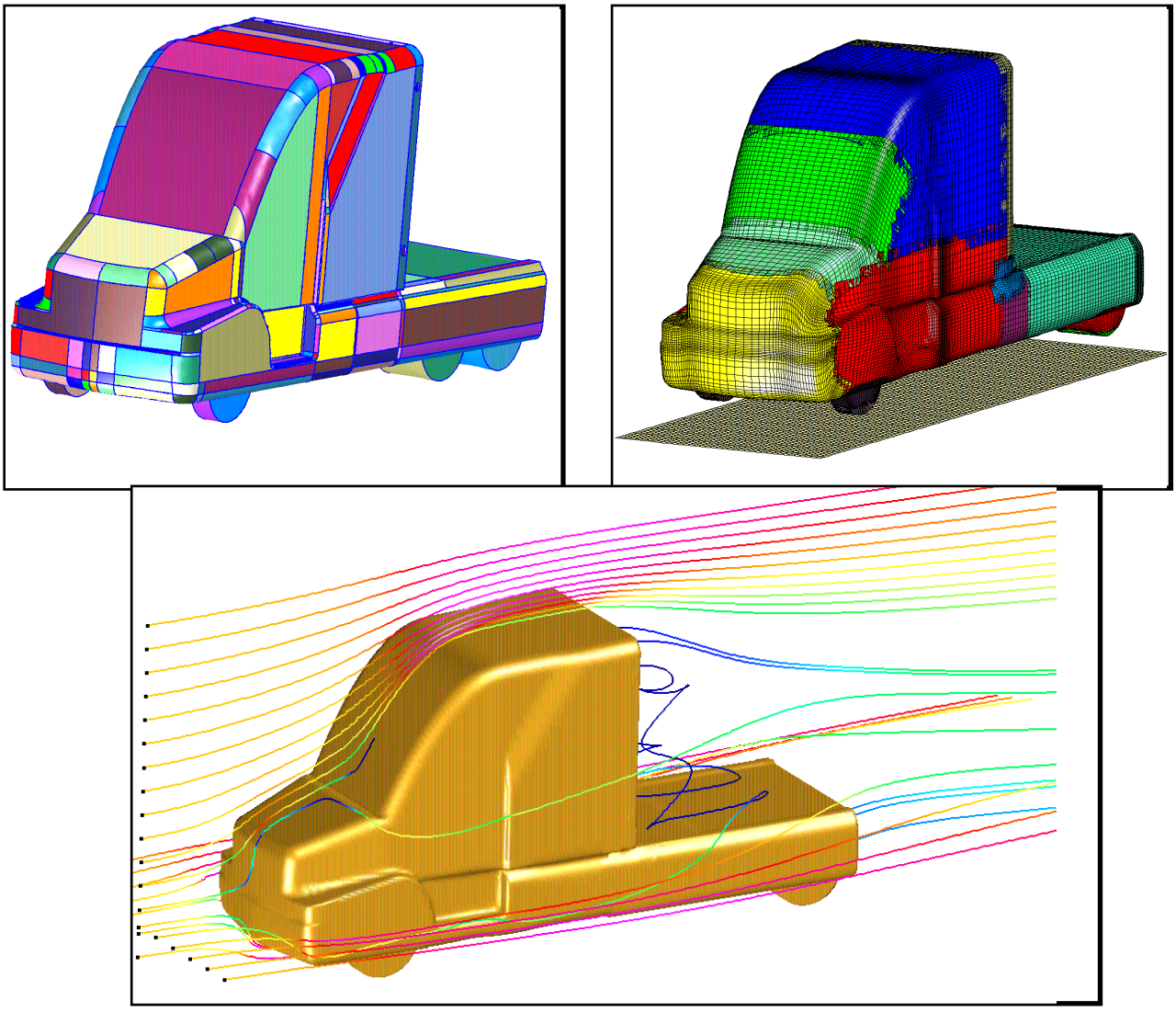
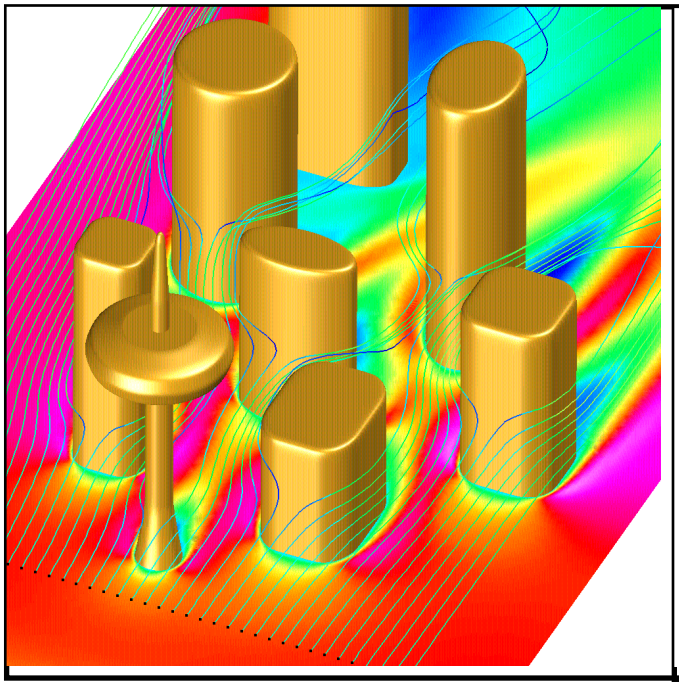


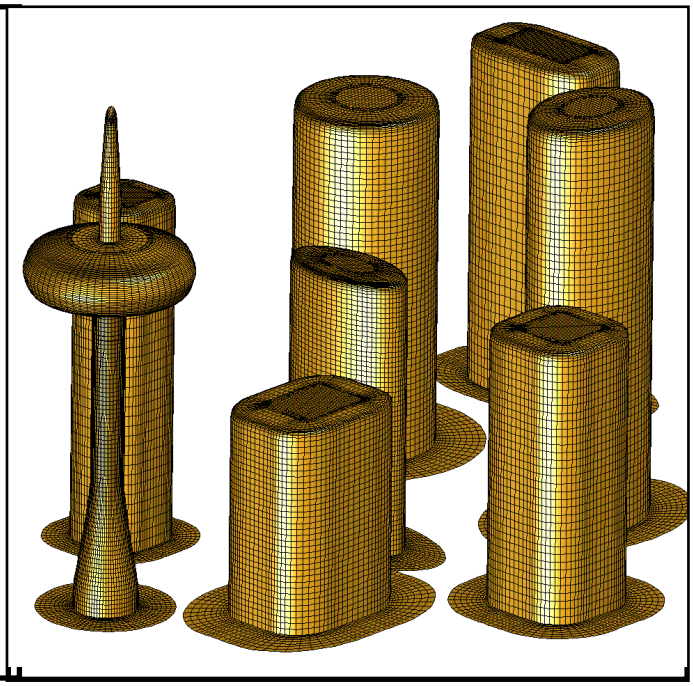
Figure 11: Incompressible flow past the cab of a truck. Shown are the CAD geometry, the grids and some tracer particles.

12.7 Incompressible flow past a city scape

Figure (12.7) shows a computation of the incompressible Navier-Stokes equations for flow past a city scape. The steady state line solver was used for this computation. The command file for generating this grid is `Overture/sampleGrids/multiBuildings.cmd` and the Cgins command file is `ins/cmd/multiBuildings.cmd`.



3D flow past a city scape.



Overlapping grid for a city scape.

Notes:

- pseudo steady-state line implicit solver, 4th-order dissipation,
- local time-stepping (spatially varying dt)
- requires 1.4GB of memory,
- cpu = 29s/step,
- 2.2 GHz Xeon, 2 GB of memory

References

[1] D. L. Brown, G. S. Chesshire, W. D. Henshaw, and D. J. Quinlan, *Overture: An object oriented software system for solving partial differential equations in serial and parallel environments*, in Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing, 1997.

[2] D. L. Brown, W. D. Henshaw, and D. J. Quinlan, *Overture: An object oriented framework for solving partial differential equations*, in Scientific Computing in Object-Oriented Parallel Environments, Springer Lecture Notes in Computer Science, **1343**, 1997, pp. 177–194.

[3] W. D. Henshaw, *A fourth-order accurate method for the incompressible Navier-Stokes equations on overlapping grids*, J. Comput. Phys., 113 (1994), pp. 13–25.

[4] ——, *Overture: An object-oriented system for solving PDEs in moving geometries on overlapping grids*, in First AFOSR Conference on Dynamic Motion CFD, June 1996, L. Sakell and D. Knight, eds., 1996, pp. 281–290.

[5] ——, *Mappings for Overture, a description of the Mapping class and documentation for many useful Mappings*, Research Report UCRL-MA-132239, Lawrence Livermore National Laboratory, 1998.

[6] ——, *Ogen: An overlapping grid generator for Overture*, Research Report UCRL-MA-132237, Lawrence Livermore National Laboratory, 1998.

[7] ——, *Oges user guide, a solver for steady state boundary value problems on overlapping grids*, Research Report UCRL-MA-132234, Lawrence Livermore National Laboratory, 1998.

[8] ——, *Plotstuff: A class for plotting stuff from Overture*, Research Report UCRL-MA-132238, Lawrence Livermore National Laboratory, 1998.

[9] ——, *Cgins user guide: An overture solver for the incompressible Navier-Stokes equations on composite overlapping grids*, Software Manual LLNL-SM-455851, Lawrence Livermore National Laboratory, 2010.

[10] W. D. Henshaw and H.-O. Kreiss, *Analysis of a difference approximation for the incompressible Navier-Stokes equations*, Research Report LA-UR-95-3536, Los Alamos National Laboratory, 1995.

[11] W. D. Henshaw, H.-O. Kreiss, and L. G. M. Reyna, *On the smallest scale for the incompressible Navier-Stokes equations*, Theoretical and Computational Fluid Dynamics, 1 (1989), pp. 65–95.

[12] ——, *Smallest scale estimates for the incompressible Navier-Stokes equations*, Arch. Rational Mech. Anal., 112 (1990), pp. 21–44.

[13] ——, *A fourth-order accurate difference approximation for the incompressible Navier-Stokes equations*, Comput. Fluids, 23 (1994), pp. 575–593.

[14] D. C. Wilcox, *Turbulence Modeling for CFD*, DCW Industries, 2000.

Index

artificial diffusion, 12
axisymmetric, 4

boundary conditions, 14

convergence results
    INS, 33

discretization
    incompressible Navier-Stokes, 6
divergence damping, 12

minimum scale, 13

pressure-poisson system, 3

time stepping, 8
    Adams predictor corrector, 8
    implicit multistep, 9, 10

variable time stepping, 11